REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	16.Oct.02	DISSERTATION

4. TITLE AND SUBTITLE
"A THREE PHASE APPROACH TO SOLVING THE BIDLINE GENERATION PROBLEM WITH AN EMPHASIS ON MITIGATING PILOT FATIGUE THROUGH CIRCADIAN RULE ENFORCEMENT"

5. FUNDING NUMBERS

6. AUTHOR(S)
MAJ WEIR JEFFERY D

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
GEORGIA INSTITUTE OF TECHNOLOGY

8. PERFORMING ORGANIZATION REPORT NUMBER
CI02-782

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS

15. NUMBER OF PAGES
97

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government

Contents

List of Tables

List of Figures

A Three Phase Approach to Solving the Bidline Generation Problem with an Emphasis on Mitigating Pilot Fatigue Through Circadian Rule Enforcement

A Thesis
Presented to
The Academic Faculty

by

Jeffery D. Weir

In Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy in Industrial and Systems Engineering

Georgia Institute of Technology
August 2002

Summary

A three phase methodology for solving the Bidline problem for airline crew scheduling is proposed. Phase 1 ensures that the all bidlines will meet a fatigue mitigating minimum rest window that will be constant throughout the entire bid period. Phase 2 ensures that most of the bidlines follow regular weekly and monthly patterns much like shiftwork in other industries that have 24-hour-a-day operations. Finally, Phase 3 creates a final monthly schedule that will minimize the number of crews needed during a bid period to cover all of the pairings, and maintain the fatigue-mitigating rest window as well as all of the quality of life issues addressed in Phase 2. Along with this methodology, a heuristic for solving Bin Packing and Cutting-Stock problems is developed.

Chapter 1

Introduction

1.1 Background

Fatigue is a familiar feeling to all of us. We experience it in various degrees throughout the day, and often are able to combat it by sleeping or taking a rest. Not all occupations have this luxury, however. The aviation industry must be able to support 24-hour-a-day operations to meet industry demands. Therefore, pilots must be able to fly at all hours of the day.

To mitigate fatigue, the Federal Aviation Administration (FAA) has set rules about how much time a pilot can fly in a 24 hour period, and how much subsequent rest a pilot must receive after flying. They have also put flying time limits at other durations as well. These rules are outlined in Federal Aviation Regulation (FAR) PARTs 121 and 135 [6, 7]. While it is the intent of these regulations to protect the public and promote safety in the airline industry, it is not known how much they actually prevent accidents and incidents. The U.S. National Transportation Safety Board (1990) "Most Wanted Transportation Safety Improvements" list currently includes a request for the study and revision of hours of service regulations in all modes of transportation. More recently, in 1995 the FAA proposed a change to FAR parts 121 and 135, citing that "The proposal would update the regulations and replace

certain out-dated regulations with a simplified regulatory approach based upon scientific studies of fatigue."[11] The airline industry balked at these new rules, and they have never been implemented by the FAA[16]. In 1999, the idea of aviation pilot flight and duty regulations and questions of fatigue again arose, this time centered around reserve crews. The House Transportation and Infrastructure Committee held hearings to gather information, but again the aviation industry presented many arguments about why the 1995 changes were not appropriate, mainly citing the lack of any safety analysis and noting that a scientific study needed to be done [17].

1.2 Problem Statement

Along with fatigue , quality of life issues seen in other workplaces are becoming more important to aircrew members. Regular workshifts, whether they be in terms of a common weekly or monthly schedule, are being sought after by pilots at many airlines. The problem for the airlines is to balance the desires of their aircrews with that of their stockholders. Can airline schedules be built that take into account these quality of life issues without impacting the overall manpower costs associated with aircrews? To further complicate the problem, some of the quality of life issues, such as fatigue, also raise concerns over public safety. Aircrew fatigue contributes to more than 21% of all errors reported in NASA's Aviation Safety Reporting System (ASRS)[14, p. 306]. Current aviation pilot duty and rest regulations are out-dated and do not apply what is now known about fatigue and the effects of our circadian rhythm. Many of these quality of life issues can be addressed by changing the way scheduling is currently done at most major airlines.

1.3 Purpose Statement

The purpose of this research is to develop a rule set and solution methodology for the bidline generation problem that addresses these quality of life issues and mitigates fatigue. This new methodology will accomplish these tasks without any significant increase in cost to the airlines, and in some instances may be able to reduce some of the costs in the planning stages of this process.

1.4 Scope

This research is not intended to be the single solution to fatigue in airline crew scheduling. It will, however, address those quality of life issues and fatigue problems that can be handled during the planning phase of crew scheduling. Fatigue induced by extended duties and shortened rests caused by weather or other unforeseen problems in the operational setting are not specifically addressed in this methodology, but the application of the rule set will certainly prevent compounding this fatigue by an inherently bad schedule. This new methodology is implemented in three phases. Phase 1 ensures that the all bidlines will meet a fatigue mitigating minimum rest window that will be constant throughout the entire bid period. Phase 2 ensures that most of the bidlines follow regular weekly and monthly patterns much like shiftwork in other industries that have 24-hour-a-day operations. Finally, Phase 3 will create a final monthly schedule that will minimize the number of crews needed during a bid period to cover all of the pairings, and maintain the fatigue-mitigating rest window as well as all of the quality of life issues addressed in Phase 2. The rule set and solution methodology take into account as many operational aspects of the commercial aviation industry as possible. Since no commercial air carrier has implemented the

rule set, it will be shown through anecdotal evidence that the rule set contains the necessary attributes to help mitigate fatigue. To determine if the solution methodology is sound, 4 test cases will be run using data from actual airline schedules. The new methodology will be compared to an actual airline schedule by coverage of flights with line crews, the amount of pilot rest, and the degree to which quality of life issues are met. Along with this solution methodology, a new technique for solving Cutting-Stock and Bin Packing problems was developed to aid in the solution of the Phase 1 problem.

1.5 Format

Chapter 2 provides a discussion on fatigue in the Airline industry and gives an explanation of airline crew scheduling in general. The duty and rest scheduling rule set and new solution methodology are presented in Chapter 3. Chapter 4 discusses implementation and test results. Chapter 5 presents conclusions and recommendations.

Chapter 2

Literature Review

2.1 Introduction

The airline industry in the United States has over 600,000 employees, and flies over 5,800 aircraft. These employees are the largest single cost to the airlines. In order to keep the labor costs down, airlines need to ensure that air crews are utilized to their fullest. It is important, however, that these crews are not over utilized to the point where fatigue can be an issue for passenger safety. This chapter first discusses circadian rhythms and other contributions to fatigue in the airline industry. Then it examines airline crew scheduling and current techniques used to build these schedules.

2.2 Fatigue

For most people and even many human factors specialists, fatigue is simply measured by how long a person does something. Often this is referred to as time on task, or is equated with the length of a work shift. While time on task is certainly a contributor to fatigue, there are many other facets of fatigue. These include, but are not limited to the time of day the work is performed, sleepiness, and work schedules.

2.2.1 Circadian Rhythm

Many of us have heard the phrase "biological clock", but few of us realize the importance it plays in our everyday life. It is the source of what is now known as our circadian (*circa* meaning around, *dies* meaning day) rhythms. These rhythms have a period of about a day, and are seen as measurable and stable daily fluctuations in physiological, psychological, and behavioral functions. These functions include body temperature, hormone secretion, heart rate, blood pressure, digestion, sensory acuity, physical and mental performance as well as many others [10]. Circadian rhythms should not be confused with "biorhythm" which was a popular theory in the 1970's, but was discredited in the early 1980's [14, p. 308].

Before the onset of artificial lighting, humans typically lived in harmony with their circadian rhythm system, working during the day and sleeping at night. This pattern is much less common today. Each year increasing numbers of people must work at times that conflict with their circadian rhythms. Pilots routinely have to fly at various times of the day or night. Additionally, the substantial irregularity of a pilot's duty hours creates more stress in comparison to those of industrial shiftworkers, as pilots are frequently adjusting their work-rest schedules.

Given the evolutionary legacy and amount to which circadian rhythmicity effects our every day life, it is not surprising that pilots have difficulty countering its influences [14]. While our body clock is more than capable of monitoring the passage of time, it is different than other clocks in that its period is flexible, and must be constantly synchronized in order to accurately predict the timing of periodic events. This synchronization is done through external cues that themselves are cyclic, usually

on a 24 hour period. In animals things like sunrise-sunset, ambient temperature, and food availability as important synchronizing events. While in humans, social time, work schedules, and group activities are more important, again partly because of the use of artificial lighting [14].

For pilots flying between different time zones, the external cues change and the circadian clock can become out of synchronization with its environment. The normal low in circadian rhythms is best estimated to be between 0200 hours and 0600 hours home base time. Research indicates that if 3 or fewer time zones are crossed, the circadian low period will not quickly adjust to the local time. If 4 or more time zones are crossed, then the circadian low period will remain the same with respect to home base time for the first 48 hours, then it will be have adjusted to 0200 hours to 0600 hours local time [10]. It is also important to note that not only is the circadian clock out of synchronization with the external cues, but also each of the circadian rhythms will be out of synchronization with each other as they all adjust differently. It is estimated that the circadian rhythms resynchronization can take several days and differs in length depending on whether the time shift is positive or negative relative to home time. In all research studies circadian rhythms adjust faster when flying west, a negative time shift, than east, regardless of what time of the day or whether the pilot was returning home or leaving home [14].

As mentioned earlier, our circadian rhythms effect when and how we sleep. Many studies have shown that when we sleep is at least as important as how long we were awake before we attempted to sleep. The body's temperature cycle greatly effects how long we sleep. Going to sleep near the temperature peak results in most REM

sleep occurring in the second half of the sleep span, where going to sleep near the low point results in a shift of the REM sleep toward the first half [14]. So, the timing of sleep can definitely affect the quality of sleep.

Circadian rhythms not only control how we sleep, but also influence when we feel sleepy. Using the Multiple Sleep Latency Test (MSLT), researchers have been able to identify two periods of time in the day when individuals are their most alert [14]. There is a peak of alertness at 0930 just after a night's sleep, and then again between 1930 and 2130. This second period of alertness comes after the mid-afternoon period, about 1530, where the body exhibits its maximal daytime sleepiness, indicating that even with no "napping", the body's biological clock has some influence on the underlying level of sleepiness we feel. While sleepiness does not necessarily equate to fatigue, aviation professionals view it as one of the most operationally significant aspect of fatigue [14].

While circadian rhythms are not the only factor contributing to pilot fatigue, it is an important factor to consider when building crew schedules. Disrupting the circadian rhythm can lead to loss of sleep, decreases in performance and alertness, and even health problems.

2.2.2 Work Schedules

Most industries that require 24 hour a day operations have developed schedules that consist of different shifts. The most common breakdown of the day is into three shifts: days, evenings and nights. While developing the shift idea was simple, deciding which shift a worker will be assigned to has been an area of intense study (see [20, p.

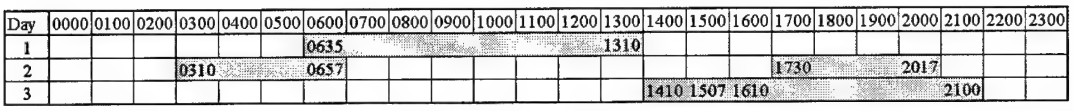

Figure 1: Pairing Example 1

1004]). Some believe in rotating each member through each shift, but then there are disagreements about how fast that rotation should be. Others recommend leaving workers on a single shift for extended periods of time, citing better performance as well as a subjective improvement in job satisfaction, health, and well being. In either case it is important to look at the risks involved with shift rotation and the hazards associated with working through a circadian low.

2.2.3 Scheduling Sources of Fatigue in the Airline Industry

As discussed earlier, aircrews have varying work-rest schedules. Figure 1 is an example of an actual airline crew pairing. It is 3 day pairing from a major U.S. air carrier that meets all FAA and union-airline negotiated rules. It consists of 5 flight legs over 3 days and has 3 rest periods. The duty periods are shown in gray with their corresponding start and end times; all times given are home base times. The rest periods then are shown as contiguous white blocks. Each rest period is at least 10 hours, with the longest being almost 18 hours. However, the aircrew will have to adjust their rest time after every duty. This is an interesting pairing, as it has 2 duties in a single day. Both the second and third duties are flown during the time of day that the crew was at rest the previous day. Similarly, the rest period between the second and third duties is at the same time of day as the crew was flying the day before. This means that a pilot will be sleeping at times when the previous day he

was required to be awake and alert, and will be flying at times when the previous day he was asleep. This type of disruption in sleep patterns definitely leads to fatigue, and can cause safety problems [20, p. 1029].

These types of disrupted sleep workshifts are not uncommon. Figure 2 is another

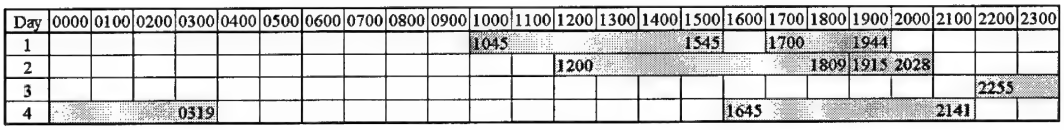

Figure 2: Pairing Example 2

example of the work-rest time shifts just talked about. In this example the first and second sleep shifts are the same but then a "red-eye" flight is added followed by a same day duty. In this case not only is the work-rest time shifted, but the final work day is shortened causing even more circadian desynchronization.

2.3 Airline Crew Scheduling

2.3.1 Terminology

For a passenger, a flight is a trip from a starting location, origin, to a destination. Often these flights are short in duration, requiring multiple flights to reach the final destination. This is also true for flight crews. An origin destination pair that is flown non-stop will be refer to as a flight leg. A duty, then, is a group of flight legs flown in sequence. Many times a flight crew will have to sit (rest) for a short period in between flight legs. At the end of a duty a flight crew will be given an overnight rest. This overnight rest may not occur during hours when it is dark outside. A crew pairing is

a sequence of duties and rest periods that starts at a crew base, lasts one to four days, and usually ends at a crew base. When a pairing does not end with a flight to the crew base, or if for their next duty a crew needs to get to an airport other than the one where they have landed, the crew will deadhead to the new location. A deadhead flight is one in which a flight crew is assigned to a flight leg as a passenger. Finally, a sequence of crew pairings is called a bidline. This bidline is the schedule a flight crew will follow for the bid period, usually a month's worth of work. It is during the crew pairing generation that cost is most affected, since during this procedure it is important that the pairings cover all of the flight legs with the minimum excess cost to the airlines.

2.3.2 Crew Pairing Generation

2.3.2.1 Legality and Rules

There are several difficulties that must be overcome when generating crew pairings. The first deals with legality of the crew pairing. As previously mentioned in Chapter 1, the FAA has rules governing the amount of time a crew can fly, and how much time they must be given for an overnight rest. Along with these rules, are the many union rules agreed to by the airlines regarding work schedules, e.g. how long a crew must be given between flights, how many flying hours a crew is allowed in a given period, maximum number of duties or days of work, etc. These rules together combine to determine the length of a duty period. Usually a duty period consists of around eight hours of flying, with a total duty duration with sits, in-briefs and de-briefs being around 12 hours.

In addition to the legality concerns for pairings, the cost structure of a flight crew is complex. The crew has a negotiated minimum hours of pay for various aspects of the pairing. If the guaranteed hours of pay is more than the actual number of hours of duty, then the crew receives extra pay known as credit. Credit is calculated as the difference between the guaranteed hours of pay and the actual hours flown and can be caused by many scheduling situations. The three most common are (1) long or frequent sits between flights, (2) long overnight rests between duties, and (3) "deadheading", usually from their last duty back to a crew base. Credit is calculated as the maximum of the several different pay guarantees.

Finally there are operational constraints to consider when generating crew pairings. First, the number of crews available at a crew base will determine the minimum and maximum number of flying hours available. Second, many airlines prefer to keep the crews on the same plane during a duty period. This is made difficult as the hub-and-spoke network of many airlines, designed to give passengers many ways to connect to other flights, also gives crews many opportunities to change planes. While it is desirable to keep crews on the same plane for a pairing, allowing them to change planes can lead to significantly better pairings. Finally, the length of the pairing has many operational implications. Long pairings can cause greater difficulty in rescheduling if weather or other factors cause them to be interrupted, while shorter pairings may require more crews at each crew base to cover all of the flights.

Adding to the complexity of the rules and legality issues is the shear number of possible pairings. As alluded to above, the number of legal pairings available due to the hub-and-spoke network of most airlines can be quite large, over 5 million even

with small fleets.

2.3.2.2 Solution Methodologies

The crew pairing problem is an instance of an important class of combinatorial optimization problems known as the *set partitioning problem.* In the *set partitioning problem* there is a set $A = \{1, 2, \ldots, m\}$ and a collection of subsets $\{A_j\}$ for $j = 1, 2, \ldots, n$. The problem then is to find a partition of A using the subsets A_j. This class of problems is readily formulated as a $0 - 1$ integer program (IP). Given costs c_j for each subset A_j and an objective to find the minimum cost partition results in the following IP

$$\min \sum_{j=1}^{n} c_j x_j$$

subject to **(SETPAR)**

$$\sum_{j=1}^{n} a_{ij} x_j = 1 \quad \text{for } i = 1, \ldots, m$$

$$x_j \in \mathcal{Z}^+ \quad \text{for } j = 1, \ldots, n$$

Where:

$n =$ the cardinality of the collection of subsets

$m =$ the cardinality of the set A

$a_{ij} =$ is 1 if A_j contains element i of set A; 0 otherwise.

$x_j =$ is 1 if A_j is used in a partition; 0 otherwise.

The correspondence between the crew pairing problem and the *set partitioning problem* is that each element of the set A is a daily flight leg, and each subset A_j is a crew pairing. The cost associated with each crew pairing is usually the minutes of credit, since the base pay of a pilot is fixed.

To solve these large scale IPs, U.S. air carriers have developed various techniques. Two of them, American Airline's TRIP (Trip Reevaluation and Improvement Program) [1] [2], and Delta's DOC (Delta Optimization Code) and KORBX will be discussed. Neither guarantee integer optimality.

In TRIP, the process is started by an initial solution, usually some adaptation of the previous month's solution. The TRIP code then iteratively selects and solves a subproblem to try and improve the current solution. The idea is to choose a set of flights, and then solve a subproblem using only those flights. This subproblem is solved to optimality and these pairings are added to the current solution set which is then re-optimized. The process is repeated in this fashion until no more improvement can be found. As with any subproblem approach, a better solution may exist even though no improvement from the current solution can be found. To try and improve the solutions found by TRIP some global techniques have been implemented [2].

Delta's approach is similar in that they too never try to solve the entire IP. Using the dual prices from the optimal solution of the Linear Programming (LP) relaxation, they find a "good", as defined by reduced cost, subset of the pairings. From this subset of pairings, again the LP relaxation is solved, and some number of the variables are fixed to 1. Once the variables are fixed either normal branch and bound techniques are used, or, if the solution is not close enough to a preset solution value (some factor times the LP relaxation solution), new pairings are generated and a branch and column generation technique is used. While this technique is proprietary, a description of a branch and price technique by Barnhart et al can be found in [3].

Most recently, in a dissertation at the Georgia Institute of Technology [13] Gopalakrishnan et al used a non-negative least squares (NNLS) approach to solve the crew pairing problem. This too is a subproblem approach. The NNLS algorithm was used in a column generation technique to solve the LP relaxation, and simple branch and bound was used to find an integer solution from some number of columns chosen by reduced cost.

2.3.3 Bidline Generation

2.3.3.1 Legality and Rules

As with crew pairings, there are legal issues to deal with concerning bidlines. All of the FAA rules concerning duty and rest in a 24 hour period still apply to bidlines since pairings may be flown back to back. And, there are also rules in FAR Part 135 and 121 that govern the total number of hours that may be flown in 7 consecutive days, a calendar month, and a calendar year. Again there are also contract negotiated rules for bidlines, e.g. maximum number of days of work per month, maximum number of consecutive days of work, minimum consecutive days off, etc. These rules combine to determine how many hours of pay and credit each bidline will contain. Typically, a bidline will consist of between 68 and 78 hours of pay and credit over 12 to 18 days of work, and use between 3 and 9 pairings.

Operational considerations also affect the bidline process, analogous to those of crew pairing generation. The number of crews available at a crew base dictate the total number of bidlines that must be generated, which in turn sets what the minimum

and maximum flying can be in the bidline. Many of the airlines have purity considerations, like the idea of crews not changing planes. For example, it may be desirable for bidlines to be built using the same pairing or trip, or to have all the blocks of work start on the same day. It may also be desirable to leave say 7 or more days of consecutive days off so that recurring training can be scheduled.

2.3.3.2 Solution Methodologies

The bidline generation problem is similar to the crew pairing generation problem. It too can be modeled using the *set partitioning problem.* In this case each element of the set A is a daily pairing, and each subset A_j corresponds to a crew schedule, i.e. bidline, for the entire bid period. The cost c_j associated with each bidline can take on many forms. It can be a penalty function determined by how well the purity rules are met, a utility function based on meeting any number of the contract negotiated minimums and maximums, or a combination of both. In many cases, the cost values for each bidline are almost identical and must be dealt with carefully as will be seen in Chapter 4.

As with crew pairing generation, each airline has its own proprietary techniques for solving the bidline generation problem. These techniques range from the same techniques used to solve the pairing generation to some newer techniques using TABU searches [8] and genetic algorithms [4].

2.4 Conclusion

Since the current solution techniques for the crew pairing and bidline generation problems only take into account FAA regulations, a new methodology is needed to further minimize or mitigate in some way aircrew fatigue. Gopalakrishnan et al demonstrated that fatigue mitigating rules can be enforced in crew pairing generation with no significant increase in cost [13]. The solution methodology developed in this thesis demonstrates that bidlines can be generated and schedules found that mitigate crew fatigue. These bidlines and final schedules meet all of the FAA regulations and airline contract negotiated rules. Chapter 3 outlines a set of fatigue mitigating rules and presents the three phase approach to solving the bidline generation problem.

Chapter 3

Methodology

3.1 Introduction

In this chapter, the circadian rule set and methodology of the three phase approach to solving the aircrew bidline generation problem is developed. Then, the model formulations and solution procedures are presented.

3.2 Circadian Rule Set

This rule set is designed to put the crew on a standard workshift like other businesses with 24-hour-a-day operations, thereby addressing as many of the scheduling and circadian rhythm influences on fatigue. First, the amount of time between pairings will be addressed. Second, the duration of the rest between duties will be determined. Finally, the timing of the rest in terms of when during the day it occurs will be looked at.

To build shifts for the crews, the time between pairings will be considered first. Two consecutive nights of usual sleep are minimally necessary to stabilize sleep patterns and return waking performance and alertness to usual levels [10]. While this could be accomplished in a 36 hour period, the rule for rest between pairings will be a minimum of 48 consecutive hours with two usual (circadian) 8 hour sleep shifts and

no flying duties. Ideally these recovery periods should cover the same days of each week to give a regular workshift; the largest pairings are usually only 3 or 4 days, so it is possible to build workshifts of 4 days on and 3 days off meeting all requirements.

When considering the time off between duties, a taxonomy for rest periods must be described. The term rest window will indicate the continuous period of time between any two duties. A rest window intersection then is the largest continuous block of time that contains no portion of a duty repeated over a given number of days. When building pairings and bidlines with a rest window rule, this is simply the intersection of the rest windows. To account for pairings built without any rest window rules, the way to calculate a rest window intersection must be altered. For example, looking back at Figure 1 the intersection of the rest windows is only 1 hour while there is a block of time from 2100 hours until 0310 hours every day where the crew has no duty. Thus this pairing has a 6 hour and 10 minute rest window intersection. To see

Table 1: Rest Windows for Pairing Example

Rest	Begin	End
1	1310	0310
2	0657	1730
3	2017	1410

this, Table 1 shows the rest periods for each duty in the pairing. By inspection, the intersection is from 1310 hours until 1410 hours. Table 2 shows that the latest end time for a duty is 2100 hours and the earliest a duty begins is 0310 hours. So there is a block of time from 2100 hours until 0310 hours every day where no duty occurs. This is in fact the largest continuous block of time with no duty.

Table 2: Duty Times for Pairing Example

Duty	Begin	End
1	0635	1310
2	0310	0657
3	1730	2017
4	1410	2100

Focusing on the rest window intersection duration and rules to build them, more than just the time spent sleeping must be considered. When a crew ends its duty for a day and goes into rest status, they must first leave the airport and get to their hotel. Depending on the time of day, they may need to eat a meal, and as discussed in Chapter 2, some time will be needed for the crew to get out of the working mindset and into one conducive for sleep. Similarly, before the crew reports for their next duty they will need to get back to the airport, eat any necessary meals and get back into the mindset of preparing for a flight. While the time to do all of these things varies widely by the individual, an hour on either end of the sleep time does not sound unreasonable. Adding this to a standard 8 hours of sleep establishes a rest window of 10 hours as the minimum number of hours for a rest window. It is important to note that while it is easy to build this duration of a rest window for a single duty, building pairings and bidlines that maintain this 10-hour rest window is much more difficult.

What needs to be addressed next is when this rest window occurs in the day. Our circadian rhythms are at their lowest typically between 0200 hours and 0600 hours home base time of every day[10]. Ideally, the rest window should encompass the entire low in circadian rhythm. This leaves a range of time from 2000 hours on the current day until 1200 hours the next day in which to fit at least 10 hours of the rest window. The

next best rule would be to try to capture a large portion of the circadian low, say 50%.

Obviously, it will not be possible to achieve this for all duties. As flying is a 24-hour-a-day operation, there are crew base and fleet combinations where flights will occur during the times of lowest circadian rhythm. These flights are often referred to as "red-eye" flights, and there are several ways to deal with them. First, every attempt will be made to schedule these flights as the last flight leg in a pairing, and this leg will not be included in the rest window intersection calculation. This way, any fatigue brought on by activity during the circadian low will not be experienced in or compounded by any subsequent flights. Instead, the "red-eye" flight will be followed by the recovery period of at least 48 hours of no duty.

This, however, will not be possible for "red-eye" flights that do not return the pilots to their crew base. For these "red-eye" flights, one of two rules will be used. First, for flight legs to destinations that do not have multiple recurring daily flights, or will arrive at a time when FAA required minimum rest times will not allow the crew to connect with a departing flight until the next day. When this happens a forced extended rest window will occur. These rest windows will automatically be larger than 10 hours, and usually exceed 24-hours. This 24-hour rest period should allow the crew to rest at the appropriate time with respect to whatever rest window intersection the other flights enforce. In these cases, the duty times of the "red-eye" flight will be ignored, and the 10-hour rest window intersection will be built with other duties in the same way. Finally, if necessary the same rules as for other duties will be used, and a regular work schedule with a minimum 10-hour rest window will be built ignoring the circadian low.

For a typical shiftworker, there is a rest window intersection of 12 to 16 hours that, depending on shift rotations, will remain constant for a long period of time, commonly a month or more. Mirroring this idea in the aviation world can be difficult as a single shift for a crew could last 12 or more hours, leaving a very narrow time in which to find a rest window intersection that meets the criteria. The idea is to form pairings with a rest window intersection that meets the criteria, and then bidlines whose rest window intersection meets the criteria. If this is accomplished, the schedules will have a block of 10 continuous hours each 24-hour period for an entire month in which the crew will never have duty scheduled.

Building these pairings and bidlines is no more difficult than building current ones. In simplest terms the rest window intersection criterion is a filter to run the pairings and bidlines through. Gopalakrishnan et al has shown that there are still a significant number of pairings remaining that meet these criteria [13]. So what is left to show is that these pairings can be combined into bidlines and a schedule that meets all of the other FAA and negotiated rules.

3.3 Three Phase Approach

This section describes a three phase approach for building aircrew schedules. The idea is to split the bidline problem into three separate, less complex optimization problems: Phase 1, Phase 2 and Phase 3. Phase 1 finds multi-sets of non-dated pairings that are feasible to form a bidline with respect to a certain rule-set and together cover the occurrences of each pairing during the bidperiod. Such a multi-set will be referred

Table 3: Rule Set for 3 Phase Solution Methodology

Phase	Rule	Variable
1,2,3	Bidlines must meet a minimum Pay and Credit	Min_{credit}
1,2,3	Bidlines cannot exceed a maximum Pay and Credit	Max_{credit}
1,2,3	Bidlines cannot exceed a maximum number of total pairings used	$Max_{pairings}$
1,2,3	Bidlines cannot exceed a maximum number of total days of work	$Max_{workdays}$
1,2,3	Bidlines must meet a minimum rest window intersection amount	Min_{rest_window}
2,3	Bidlines cannot exceed a maximum number of flying hours in a 7 day window	Max_{fly}
2,3	There can be no single days off	
2,3	A bidline cannot work more than 5 consecutive days	

to as a "pattern". The rule set for Phase 1 consists of only those rules which do not depend upon the start and the end date of a pairing, e.g. the minimum credit that a bidline must satisfy. Phase 2 takes the patterns chosen in Phase 1 and uses them to generate bidlines as follows. Each non-dated pairing in a pattern is substituted by its set of dated pairings so that each pattern yields its corresponding set of dated pairings. Bidlines are generated from these sets of dated pairings with respect to all the rules that were not enforced in Phase 1. Phase 2 tries to cover every dated pairing using "good" bidlines. Finally, Phase 3 tries to cover only those dated pairings not covered by Phase 2. In Phase 3 all possible patterns are used, but only those dated pairings not covered in Phase 2 are used.

Table 3 shows all of the rules and the phase of the problem in which they are enforced. Along with each rule is a variable that will be referenced throughout the remaining chapters. Chapter 4 will discuss the effect of these variables.

3.3.1 Phase 1

This section presents the Phase 1 mathematical formulation, and solution methodologies. First, the Phase 1 problem is discussed. This is followed by the description of the Phase 1 mixed integer problem (MIP). Finally, the Phase 1 MIP solution algorithm is discussed.

3.3.1.1 Problem Description

Simply stated, the Phase 1 problem is to find a minimum a set of patterns that covers all of the pairings to be flown for the bid period. When talking about patterns there are two measures used, the size of the pattern, i.e. the total number of pairings in the pattern, and the number of unique pairings used in the pattern. For example, this pattern

$$\begin{bmatrix} 3 & 3 & 3 & 3 & 8 & 8 & 8 \end{bmatrix}^T$$

has a size of 7 pairings, but uses two unique pairings, pairing 3 and pairing 8. The Phase 1 problem can be represented as

$$\min \sum_{j=1}^{n} c_j x_j$$

subject to **(IP 1)**

$$\sum_{j=1}^{n} P_{ij} x_j = D_i \quad \text{for } i = 1, \ldots, m$$

$$x_j \in \mathcal{Z}^+ \quad \text{for } j = 1, \ldots, n$$

Where:

n = the number of patterns generated

m = the number of pairings

c_j = the cost of using pattern j.

P_{ij} = the number of times pattern i uses pairing j.

D_i = the number of times paring i is flown in the bid period

x_j = the number of times pattern j is chosen

For this research, each column is a pattern of pairings that meets the following rules: 1. the sum of the credit of the pairings in the pattern is greater than or equal to the minimum pay and credit, 2. the sum of the credit of the pairings in the pattern is less than or equal to the negotiated maximum pay and credit, 3. the number of days worked is less than or equal to the maximum number of negotiated work days, 4. the total number of pairings in the pattern is less than or equal to the maximum number of pairings to be used in a bidperiod, 5. the pattern must have a valid rest window intersection. There is a row for each pairing flown during the bid period. Using this formulation, the Phase 1 problem is similar to the bin packing problem [5, p. 195].

3.3.1.2 The Bin Packing Problem

The bin packing problem is well studied in integer programming. In one dimension, there are a number of items with the same characteristic, say weight. These items must be placed in bins for holding. The bins have a limited capacity based on the characteristic being measured. The goal is to use as few bins as possible to hold all of the items. For this representation, a pattern then tells which items are being placed together in a bin. The bin packing problem is formulated exactly as **IP 1**. Each row represents a different size of an item, and $c_j = 1$ for all j in order to minimize the total number of bins used. Also, each pattern need only satisfy one rule: the sum of

the measurements of the items must be less than or equal to the capacity of the bins. Since the number of patterns can be quite large, the bin packing problem has been solved in the past using delayed column generation[5, p.198].

The LP relaxation of **IP 1** is solved with a set of starting patterns. Next, a column can be generated by solving the knapsack problem as given by **IP 2** using the dual values from a current solution as the objective values for each item size.

$$\max \sum_{j=1}^{n} \mu_j a_j$$

subject to **(IP 2)**

$$\sum_{j=1}^{n} w_j a_j \leq Capacity$$

$$a_j \in \mathcal{Z}^+$$

Where:

n = the number of different size items.

μ_j = the dual value from the current solution for item j.

a_j = the number of times item j is placed in the bin.

w_j = the weight or other characteristic of item j.

Since the bin packing column generation problem has only a single constraint, i.e. a knapsack problem, it can be solved efficiently, for small right hand sides, using dynamic programming[18, p. 433]. The column generation problem for the Phase 1 problem, however, has many constraints that a pattern must satisfy, and a single objective knapsack problem will not work. The column generation technique for Phase

1 is discussed in Section 3.3.1.3. Not only is dynamic programming efficient, but it allows for more than just the optimal pattern to be found. At this point either the column with the highest reduced cost, or any number of columns that have a good reduced cost, can be added to the problem. Then the LP relaxation is resolved. This process repeats until an optimal LP solution is found. However, passing from an optimal fractional valued LP solution to an optimal integer valued solution is not easy [5].

The most direct way is to use branch and bound techniques which can be found in all commercial solvers. However, these techniques tend to be very slow for large problems, and cannot be guaranteed to find an optimal solution given only a portion of columns of the original problem, as happens when using delayed column generation. Many heuristics have been developed to solve this problem. One method, the first-fit decreasing algorithm has a guaranteed performance that if n bins are needed in the optimal integer valued solution then at most $\frac{11}{9}n + 4$ bins will be used by its solution [5].

When the righthand sides of the bin packing problem get large, the bin packing problem is known as the cutting-stock problem. This problem arose from the manufacture of materials such as paper, textiles, metallic foils and the like. Large rolls called raws are cut into smaller rolls known as finals. The cutting is done on machines by knives that slice the raws into finals in much the same way that bread is sliced. In the cutting process, there is a cost associated with changing the knives on the cutting machines to begin a new pattern. To try and minimize this cost, a best optimal solution will use each pattern as many times as possible. While this is not a requirement for the bin packing problem, it does lead to a heuristic that can solve

Table 4: All Maximal Patterns for Bin Packing Example

Final Widths	Maximal Patterns																								
8	0	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	4	5	6
10	0	0	1	1	2	3	5	0	0	1	2	3	4	0	1	1	2	3	0	0	1	2	0	1	0
12	2	4	0	3	2	0	0	0	3	1	0	1	0	1	0	2	1	0	0	2	1	0	1	0	0
20	1	0	2	0	0	1	0	2	0	1	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0
Total Widths	44	48	50	46	44	50	50	48	44	50	48	50	48	48	46	50	48	46	44	48	46	44	44	50	48

the Phase 1 problem and other similar bin packing problems.

The following example will be used to illustrate this heuristic. Let a bin have a capacity of 50, with 50 items of weight 8, 60 items of weight 10, 28 items of weight 12 and 19 items of weight 20. For a problem of this size, all patterns (bin packings) that are maximal is size can be enumerated. Table 4 shows every maximal pattern for the example. While there are other combinations of the final widths, all others either exceeded the capacity of a bin, or have enough slack to allow another item to be placed in a bin. While the patterns do not have to be maximal in terms of weight, a minimum capacity will be enforced, as it leads to better solutions.

This heuristic builds maximal columns for every pattern. A maximal column is a column that represents using pattern j, k_j times. A new column is added to **IP 1** with $P_{i(n+j)} = k_j P_{ij}$, where j is the index of the column, and n is the total number of patterns in the original problem. The objective value coefficient must also be adjusted. The new maximal column will have a cost

$$c_{n+j} = BONUS_{n+j}(c_j).$$

This new maximal column will be a $0-1$ integer variable instead of a general integer, and the $BONUS$ term will be used to direct the solution. If $BONUS_{n+j}$ is anything other than k_j, a constraint of the form

$$\sum_{j=1}^{n}\Big(x_j + k_{n+j}x_{n+j}\Big) - Diff = \lceil (LP_{obj}) \rceil. \tag{1}$$

must be added to ensure the minimum number of patterns are used. Here LP_{obj} is the optimal objective value of the LP relaxation of the original problem without maximal columns. $Diff$ is a variable to capture how much the IP solution is different than the LP solution, and has a big M penalty associated with it in the objective function. So, the initial heuristic IP is given by **IP 3**.

$$\min \sum_{j=1}^{n}\Big(c_j x_j + BONUS_{n+j}x_{n+j}\Big) + M * Diff$$

subject to

$$\begin{aligned}
&\sum_{j=1}^{n} P_{ij}x_j = D_i \quad \text{for } i = 1,\ldots,m \\
&\sum_{j=1}^{n}\Big(x_j + k_{n+j}x_{n+j}\Big) - Diff = \lceil (LP_{obj}) \rceil \qquad \textbf{(IP 3)} \\
&x_j, Diff \in \mathcal{Z}^+ \quad \text{for } j = 1,\ldots,n \\
&x_{n+j} \in \mathcal{B}^+ \quad \text{for } j = 1,\ldots,n
\end{aligned}$$

Since every original pattern is maximal with respect to the capacity of a bin, it makes sense to build maximal columns for all of them. The simplest way of calculating the value of k_j is to use the following equation:

$$\inf_{\substack{i,j \\ P_{ij}>0}} \left\{ \left\lfloor \frac{D_i}{P_{ij}} \right\rfloor \right\}. \tag{2}$$

Table 5: All Maximal Columns for Bin Packing Example

Final Widths	Maximal Columns																							
8	0	0	0	0	0	0	0	9	9	19	19	20	15	38	38	28	50	40	48	42	48	48	48	50	48
10	0	0	9	9	28	57	60	0	0	19	38	60	60	0	19	14	50	60	0	0	16	32	0	10	0
12	28	28	0	27	28	0	0	0	27	19	0	20	0	19	0	28	25	0	0	28	16	0	12	0	0
20	14	0	18	0	0	19	0	18	0	19	19	0	0	19	19	0	0	0	16	0	0	0	0	0	0
k	14	7	9	9	14	19	12	9	9	19	19	20	15	19	19	14	25	20	16	14	16	16	12	10	8

However, there may be some physical information about the problem that would give us a better bound. For example, in the cutting-stock problem, if a machine must be adjusted after 50 raws have been cut, then k_j can be set as the minimum of equation (2) and 50. Further, it may also be beneficial not to choose k_j at its maximal value at all. The problem itself may show a relation between the values of P_{ij} and D_i. In this case you would want to choose k_j that exploits that relation. For example, in the Phase 1 problem for a month with 30 days, many of the righthand sides are 30. By choosing a maximal value of 5 for k_j it is more likely that a combination of maximal columns will sum to 30. If 7 is chosen, then there is no combination of maximal patterns that will sum to 30, and the remainder of the righthand side will have to be filled in by one or two non-maximal columns.

Solving the example problem using equation (2) to calculate the k_j's results in the maximal columns shown in Table 5. Using CPLEX version 6.5 with its default settings, the solution to **IP 1** using the patterns in Table 4 is shown in Table 6. This solution uses 11 different patterns, and a total of 35 bins. Adding maximal columns for only those patterns which use a whole bin, CPLEX returns the solution in Table 7 While none of the maximal columns were chosen, the negative coefficients in

Table 6: Solution Values for Bin Packing Example

Variable	Value
x2	1
x3	9
x5	2
x7	7
x9	1
x10	1
x16	5
x20	3
x22	1
x24	4
x25	1

Table 7: Solution Values for Bin Packing Example With Maximal Columns for Full Bin Patterns

Variable	Value
x1	1
x2	5
x3	9
x5	3
x7	7
x24	10

Table 8: Solution Values for Bin Packing Example With All Maximal Columns

Variable	Value
x18	1
x27	1
x31	1
x50	1

$\Rightarrow$

Variable	Value
x18	1
x2	7
x6	19
x25	8

the objective function altered the search strategy for CPLEX, allowing it to find yet another optimal solution, this time using only 6 different patterns and a total of 35 bins. Finally, the solution to **IP 3** which has all of the maximal columns is shown in Table 8. This is the best optimal solution using a total of 35 raws but only 4 different patterns. For this example, all of the columns were used. In real world problems, the delayed column generation technique will only generate some of the columns. The columns generated will be solved to an optimal LP solution, but there is no guarantee that the heuristic acting on only the columns from column generation will be able to find an optimal solution. This heuristic was used to solve the Phase 1 problem, and results from both those solutions and the solutions of other bin packing problems will be presented in Chapter 4.

3.3.1.3 Solution Procedures

Since the Phase 1 problem is the same as the bin packing problem, some of the same solution techniques can be used, the first being delayed column generation. Problem **COLGEN** shows a column generation IP problem for Phase 1. First, the rest window intersection rule enforcing constraints will be discussed.

Usually, the rest window starts in the evening of one day and ends in the morning of the next. Changing the way the times are written makes find the rest window intersection easier. Let 0000 hours of the beginning day of rest be time 0. Then simply counting the minutes, 48 hours later would be time 2880 or 2,880 minutes later. Using this time notation, a usual rest window would start before time 1440 (midnight crew base time) and end some time after it. However, for "red-eye" flights and even some late ending pairings, it is possible for a rest window to start after midnight, and end before midnight the next day. In these cases, both the start and stop times of the rest window will be less than 1440, or greater than 1440. For example, if a rest window went from 0100 hours until 1300 hours it could either be changed to 60 until 780 or 1500 until 2220. When this happens, for completeness, the pairing is duplicated, one with each representation of the rest window. In the IP description, a prime symbol (') is used to indicate the duplicates of the pairings with these types of rest windows.

COLGEN

$$\max \sum_{j=1}^{n} \mu_j (s_j + s'_j) \tag{3}$$

subject to

$$\sum_{j=1}^{n} (s_j + s'_j) \leq Max_{pairings} \tag{4}$$

$$\sum_{j=1}^{n} d_j (s_j + s'_j) \leq Max_{workdays} \tag{5}$$

$$\sum_{j=1}^{n} c_j (s_j + s'_j) \leq Max_{credit} \tag{6}$$

$$\sum_{j=1}^{n} c_j (s_j + s'_j) \geq Min_{credit} \tag{7}$$

$$y_j - \frac{s_j}{b_j} \geq 0 \quad \forall j \tag{8}$$

$$y'_j - \frac{s'_j}{b_j} \geq 0 \quad \forall j \tag{9}$$

$$y_j + y'_j \leq 1 \quad \forall j \tag{10}$$

$$start_j y_j + start_j y'_j \leq Rest_{start} \quad \forall j \tag{11}$$

$$end_j y_j + end_j y'_j \geq Rest_{end} \quad \forall j \tag{12}$$

$$Rest_{end} - Rest_{start} \geq Min_{rest\_window} \tag{13}$$

$$0 \leq s_j \leq b_j \tag{14}$$

$$0 \leq s'_j \leq b_j \tag{15}$$

$$s_j, s'_j, Rest_{start}, Rest_{end} \in \mathcal{Z}^+ \tag{16}$$

$$y_j, y'_j \in \mathcal{B}^+ \tag{17}$$

Where:

n = the number of pairings.

μ_j = the dual value for pairing j from the current solution.

d_j = the number of days of work in pairing j.

c_j = the pay and credit in minutes for pairing j.

b_j = the number of times pairing j is used in the bid period.

$start_j$ = the start time for the rest period of pairing j.

$start'_j$ = the start time for the rest period of pairing j.

end_j = the end time for the rest period of pairing j.

end'_j = the end time for the rest period of pairing j.

$Rest_{start}$ = the start time for the rest period of the pattern

$Rest_{end}$ = the end time for the rest period of the pattern

s_j = the cardinality of pairing j in the pattern.

s'_j = the cardinality of pairing j in the pattern.

y_j = 1 if pairing j is in the pattern, 0 o.w.

y'_j = 1 if pairing j is in the pattern, 0 o.w.

Since the objective is to find the best column to enter the basis, equation (3) tries to maximize the reduced cost of the pattern. Equations (4) - (7) enforce the rules agreed upon through negotiations or FAA regulations. Equations (8) - (13) are used to determine the rest window intersection for the pattern. Equations (8) - (9) force indicator variables that show which pairings are in the pattern. Equation (10) ensures that only one instance of any pairing that has been duplicated is used. Equation (13) requires the rest window to be at least the minimum required length. Equations (11) and (12) determine the start and end time of the patterns rest window intersection.

The final 4 constraints bound the number of times a pairing can be used in a pattern, and also indicate whether variables are binary or general integers.

Given n pairings, there are at most $2n$ general integer pairings with an upper bound less than or equal to $Max_{pairings}$, $2n$ binary variables, and $5n + 5$ constraints. While this problem looks daunting, for a typical crew scheduling problem, the number of pairings n from the optimal solution of the crew pairing problem is less than 50, and $Max_{pairings}$ is less than 10. The total number of variables even with all pairings being duplicated then is less than 200 with 255 constraints. Commercial solvers have no problem finding an optimal integer solution to this problem.

Furthermore, Johnson and Barnes [15] have shown that even this multi-objective knapsack problem can be solved using dynamic programming. In this case, each objective forms a dimension of the dynamic program array. The more objectives there are the larger the dimension of the dynamic programming array. A discussion of the dynamic programming implementation is in Chapter 4.

Once a column generation scheme has been used to solve the LP relaxation of the Phase 1 problem, an integer solution to the problem must be found. To do this the heuristic discussed in section 3.3.1.2 will be used. Equations (18) - (22) show the final Phase 1 MIP. Equation (19) ensures every pairing is covered. Equation (20) ensures that no more patterns are used than there are crews available. Equation (21) is the amount of scheduled pairings, reserve crews are allowed to fly. And, equation (22) puts an upper bound on the number of times pattern j is used.

The problem may not even be feasible. The infeasibility question is handled in two ways. First, reserve crews can be used. These crews are not given a regular schedule to fly. They fill in when scheduled crews cannot meet their scheduled commitments due to delays causing extended flying and therefore extended rest, illness, etc. Most airlines have an amount of infeasibility in terms of pay and credit that they will allow not to be scheduled for line crews. The assumption is that reserve crews will fly the pairings not assigned to line crews during the bid period. Equation (21) represents this way of dealing with infeasibilities. Second, artificial variables are used for each pairing and a Big M method tries to reduce the amount of infeasibility if any exists.

Phase 1 MIP

$$\min \sum_{j=1}^{n}(w_j - c_j x_j) + \sum_{i=1}^{m} M_1 y_k + \sum_{i=1}^{m} M_2 z_k \tag{18}$$

subject to

$$\sum_{j=1}^{n} P_{ij} w_j + \sum_{j=1}^{n} MP_{ij} x_j + y_i + z_i = D_i \quad \text{for } i = 1, \ldots, m \tag{19}$$

$$\sum_{j=1}^{n}(w_j + c_j x_j) \leq LC \tag{20}$$

$$\sum_{i=1}^{m} PC_i y_i \leq RPC \tag{21}$$

$$w_j + c_j x_j \leq c_j \quad \text{for } j = 1, \ldots, n \tag{22}$$

$$w_j \in \mathcal{Z}^+ \quad \text{for } j = 1, \ldots, n$$

$$x_j \in \{0, 1\} \quad \text{for } j = 1, \ldots, n$$

$$y_i \in \mathcal{Z}^+ \quad \text{for } i = 1, \ldots, m$$

$$z_i \in \mathcal{Z}^+ \quad \text{for } i = 1, \ldots, m$$

Where:

n = the number of patterns generated

m = the number of pairings

w_j = the number of times pattern j is chosen

x_j = 1, if maximal pattern j is chosen. 0, o.w.

y_i = the number of times pairing i is covered by a reserve crew

z_i = the number of times pairing i is not covered

c_j = the number of times a pattern was combined to form maximal pattern j

M_1 = the penalty for allowing a pairing to be covered by a reserve crew

M_2 = the penalty for not covering a pairing

P_{ij} = the number of times pattern j uses pairing i.

MP_{ij} = the number of times maximal pattern j uses pairing i.

D_i = the number of times paring i is flown in the bidperiod

LC = the number of line crews available for the bid period

PC_k = the amount of pay and credit in minutes for pairing k

RPC = the amount of pay and credit in minutes allowed to be not scheduled for line crews for the bidperiod

The Phase 1 solution algorithm has gone through many changes, with two techniques working equally well. The differences between the techniques deals mainly with user preferences regarding purity. As was discussed in Chapter 2, many airlines have purity rules regarding the types of schedules desired. Two types of purity that seem to hold for most airlines are trip purity and day purity. First, some definitions are needed.

Day	Pairing 1		Pairing 2		Pairing 3	
	Orig.	Dest.	Orig.	Dest.	Orig.	Dest.
1	LAX	ATL	LAX	ATL	LAX	ATL
	ATL	BNA			ATL	BNA
	BNA	ATL			BNA	ATL
	ATL	SAV	ATL	SAV	ATL	SAV
	Rest at SAV		Rest at SAV		Rest at SAV	
2	SAV	ATL	SAV	ATL	SAV	ATL
	ATL	PHX	ATL	PHX	ATL	PHX
	Rest at PHX		Rest at PHX		Rest at PHX	
3	PHX	ATL	PHX	ATL	Deadhead PHX to ATL	
	ATL	MIA	ATL	MIA	ATL	MIA
	Rest at FLL		Rest at FLL		Rest at FLL	
4	FLL	TPA	FLL	TPA	FLL	TPA
	TPA	LAX	TPA	LAX	TPA	LAX
	Begin days off		Begin days off		Begin days off	

Figure 3: Pairing Similarity Example

A trip will be defined as the pairings between any two continuous periods of days off. A period of days off will consist of two or more consecutive 24-hour periods of no duty, each assumed to contain an 8-hour rest period. So, a trip could be a single 4 day pairing, or it could be two 2 day pairings flown with no days off between them.

Then a trip pure schedule would mean that every trip flown during the bid period is the same. Pairings will be considered similar if their rest windows occur at the same geographic location and on the same day of the pairing. For example, the pairings in Figure 3 would be considered similar. They do not fly all of the same flights, but end up every rest window in the same geographic location and fly most of the same flight legs. Bidlines that included any combination of these pairings would be considered trip pure.

Day purity deals only with what day the trip starts, and not where it goes. A day pure bidline has all of its trips start on the same day of the week. This combined with the rest window intersection rules, builds a regular work schedule comparable to shiftwork. Day purity is not handled in Phase 1 since the days a pairing will be flown on are not considered. However, depending on how important trip purity rules are determines which solution technique to use to solve the Phase 1 problem.

If trip purity is important, then all patterns that use 1 unique pairing are generated. These patterns will not be used to solve **Phase 1 MIP**. Since these patterns are all trip pure, they will be passed on to Phase 2 along with the other patterns chosen by solving **Phase 1 MIP**. To solve **Phase 1 MIP** the following algorithm is used:

Phase 1 MIP Solution Algorithm

Initialize **Phase 1 MIP** with all patterns using exactly 2 unique pairings.

Solve LP relaxation of **Phase 1 MIP**

Initialize *num_unique_pairings* to 3.

While LP relaxation solution not optimal.

While columns built using *num_unique_pairings* price out to enter basis.

Use column generation to solve the LP relaxation of **Phase 1 MIP**

num_unique_pairings + +

Generate cutting planes for maximal column binary variables.

Change objective coefficients and employ maximal column heuristic.

All patterns using a single unique pairing, and all patterns found in the optimal solution of **Phase 1 MIP** will be passed to Phase 2.

If trip purity is not that important, then all single unique pairing patterns are not forced on to Phase 2. Instead, the same algorithm as above is used except the problem is initialized with all patterns using less than 3 unique pairings.

Just before the maximal column heuristic is invoked, a number of cutting planes can be generated based on the coefficients of the maximal columns in each row and the corresponding righthand side. Many of the coefficients are within a factor of 6 of the righthand side. The magnitude of the coefficients limits the total number of maximal columns that can be positive. While not all possible limiting cutting planes are generated at this step, one cutting plane is generated for every row for each factor from 1 to 6. Section 4.1.1 and equation (23) give a mathematical description of these cuts.

3.3.2 Phase 2

This section presents the Phase 2 mathematical formulation, and solution methodologies. First, the Phase 2 problem is discussed. This is followed by the description of the Phase 2 MIP. Finally, the Phase 2 MIP solution algorithm is discussed.

3.3.2.1 Problem Description

The Phase 2 problem takes the patterns generated from Phase 1 and builds feasible dated bidlines. By using only the patterns from Phase 1, the solution space that will be searched is significantly reduced. Since the patterns chosen from Phase 1 provide a

good non-dated solution to covering all of the pairings, the hope is that these patterns can be dated and coverage will remain high.

As with Phase 1, while complete coverage is desired, it may not be feasible. The same kind of reserve crew pay and credit constraint and artificial variables will be used to handle any infeasibilities. **Phase 2 MIP** shows the set partitioning formulation of this problem.

Phase 2 MIP

$$\min \sum_{j=1}^{n} c_j x_j + M_i y_i$$

subject to

$$\sum_{j=1}^{n} B_{ij} x_j + y_i = 1 \quad \text{for } i = 1, \ldots, m$$

$$\sum_{i=1}^{m} PC_i y_i \leq RPC$$

$$x_j \in \mathcal{B}^+ \quad \text{for } j = 1, \ldots, n$$

$$y_i \geq 0 \quad \text{for } i = 1, \ldots, m$$

Where:

n = the number of bidlines generated

m = the number of dated pairings

c_j = the utility of using bidline j in the schedule.

B_{ij} = 1 if dated pairing i is used in bidline j, 0 o.w.

PC_i = the pay and credit for dated pairing i.

RPC = the amount of pay and credit in minutes allowed to be not scheduled for line crews for the bidperiod

x_j = 1 if bidline j is chosen, 0 o.w.

y_i = 1 if dated pairing i is not scheduled to a line crew, 0 o.w.

This problem can also be thought of in terms of graph theory. First, let every dated pairing represent a node in a directed acyclic graph (DAG), and then add arcs between nodes if they can legally follow each other in a bidline. Then, each bidline would represent a path in this graph, and an optimal solution would be a set of vertex disjoint paths. Another way is to look at the conflict graph generated by the bidlines. Each bidline represents a node, and then an arc will exist between two nodes if they both use the same dated pairing. In this representation, each dated pairing forms a clique, and a node can belong to as many as $Max_{pairings}$ cliques. These representations will both be discussed again in section 3.3.2.2.

During Phase 2 only bidlines that satisfy the purity rules will be considered. The number of bidlines generated then at this stage will greatly depend on the purity rules. Since dated pairings are used now, the day purity rule can be enforced. Those bidlines that satisfy both trip and day purity rules will be considered "perfect" bidlines.

3.3.2.2 Solution Procedures

There are many solution techniques to solve the *set partitioning problem.* And, while the problem is much smaller compared to looking at all dated bidlines, there are still enough bidlines that even the best commercial solvers cannot find an optimal solution

to the problem quickly, results will be discussed in Chapter 4. Again, a delayed column generation technique will be used here as in Phase 1. Depending on the number of bidlines generated, they can be stored directly or re-generated at each iteration of the delayed column generation.

As with Phase 1 first a set of good columns is needed to start the delayed column generation. Again, enumeration provides a quick way of doing this. The following algorithm is used to generate the initial set of columns for Phase 2.

Algorithm to find initial column set

Sort patterns by unique pairings smallest to largest.
Enumerate all feasible bidlines.
While perfect bidlines remain not set to 0.
    Choose first perfect bidline available and set to 1.
    Set all conflicted bidlines to 0.
While trip or day pure bidlines remain not set to 0.
    Choose first trip or day pure bidline and set to 1.
    Set all conflicted bidlines to 0.
Add all bidlines with a conflict number of 1 to all bidlines chosen.
Solve Phase 2 MIP with these bidlines only.

In the algorithm to find an initial set of columns, a conflicted bidline is one where any of its dated pairings is covered by the previously chosen bidlines. The conflict number of a bidline is the number of dated pairings in the bidline not covered by any chosen bidline. A conflict number of 1 indicates that exactly one of the dated pairings of the

bidline is not covered by any chosen bidlines. Once the optimal solution has been found, those columns are used to start the delayed column generation for **Phase 2 MIP**.

This makes a good set of initial columns for several reasons. First, all of the while loops are quick, simple depth first searches and take time $O(n)$. Second, by putting all patterns with 1 unique pairing up front, once a perfect bidline is chosen, by virtue of the enumeration scheme, the next available perfect bidline from that pattern will be a day shifted version of the previous one. Third, this guarantees a large number of perfect bidlines in the final schedule. Finally, in practice these columns form at least a 50% solution and significantly reduce the amount of time spent in the delayed column generation portion of the Phase 2 Algorithm.

Once a starting set of columns has been found the delayed column generation algorithm is used to solve the LP relaxation of **Phase 2 MIP**. The set partitioning problem though posses some difficulties for a sub-problem approach. These difficulties will be discussed in Chapter 4. Finally, once a set of bidlines that optimally solves the LP relaxation has been found, an integer solution to the problem is needed. Here, a clique based fixing heuristic is used.

Using the conflict graph of the problem, each row then is a clique in that graph. If a variable in a clique is fixed, then all other variables in that clique are set to 0. In addition, any other cliques to which this bidline belongs are also set to 0. If a bidline in a large clique is chosen, it is possible that another row is a sub clique, and then that pairing is guaranteed to not be covered. To avoid this, the cliques are sorted by

size, and a variable is fixed to 1 in the smallest remaining clique. The variable to be fixed at 1 can be chosen in several ways. The largest valued variable from the current LP relaxation solution can be chosen, or maybe only if it exceeds a minimum value, such as 0.5. Perhaps, only certain types of bidlines should be set to 1 in this process, say only perfect bidlines, or only bidlines that meet at least one purity rule. Different fixing rules will be discussed in Chapter 4, but the basic heuristic is shown below.

Phase 2 MIP Clique Based Fixing Heuristic

Use column generation to find optimal solution to LP relaxation of **Phase 2 MIP**.

Sort cliques by size smallest to largest.

Initialize set $\mathcal{S}$ to contain sorted cliques.

For each bidline determine number of cliques to which it belongs.

While $\mathcal{S} \neq \emptyset$

Find smallest clique that has a bidline that meets the fixing rules.

Fix bidline to 1.

Perform any necessary operations on $\mathcal{S}$ based on fixing rules.

Resolve LP relaxation.

Solve to optimality **Phase 2 MIP** with fixed variables.

The necessary operations to be performed on $\mathcal{S}$ fall into two categories. First, once a bidline has been fixed to 1, all cliques that it is a member of will be removed from $\mathcal{S}$ as those rows in **Phase 2 MIP** are tight. Second, fixing a bidline to 1 also fixes many bidlines to 0. It can turn out that a clique's members not set to 0 no longer meet any of the fixing rules, and that a clique can be removed from $\mathcal{S}$.

Once an optimal solution to **Phase 2 MIP** is found using the clique based fixing heuristic, a determination is made as to whether or not Phase 3 is necessary. If The solution to **Phase 2 MIP** meets the requirements for a schedule then Phase 3 is not necessary, otherwise all bidlines that are perfect or are day pure are fixed and Phase 3 will begin.

3.3.3 Phase 3

This section presents the Phase 3 mathematical formulation, and solution methodology. First, the Phase 3 problem is discussed. This is followed the description of the Phase 3 mixed MIP. Finally, the Phase 3 MIP solution algorithm is discussed.

3.3.3.1 Problem Description

Phase 3 is completely different than Phase 1 and Phase 2. In Phase 3 the entire problem is solved at once. This is possible now as the fixed bidlines from Phase 2 significantly reduce the size of the problem. The phase 3 MIP then is exactly the same as Phase 2 except it does not have as many pairings that need to be covered. **Phase 3 MIP** shows the formulation used.

Phase 3 MIP

$$\min \sum_{f=1}^{F} x_f + \sum_{j=1}^{n} c_j x_j + M_i y_i$$

subject to

$$\sum_{f=1}^{F} B_{if} x_f + \sum_{j=1}^{n} B_{ij} x_j + y_i = 1 \quad \text{for } i = 1, \ldots, m$$

$$\sum_{i=1}^{m} PC_i y_i \leq RPC$$

$$x_j \in \mathcal{B}^+ \quad \text{for } j = 1, \ldots, n$$

$$x_f = 1 \quad \text{for } f = 1, \ldots, F$$

$$y_i \geq 0 \quad \text{for } i = 1, \ldots, m$$

Where:

n = the number of bidlines generated

m = the number of dated pairings

c_j = the utility of using bidline j in the schedule.

B_{ij} = 1 if dated pairing i is used in bidline j, 0 o.w.

B_{if} = 1 if dated pairing i is used in bidline f, 0 o.w.

PC_i = the pay and credit for dated pairing i.

RPC = the amount of pay and credit in minutes allowed to be not scheduled for line crews for the bidperiod

x_j = 1 if bidline j is chosen, 0 o.w.

x_f = 1, and are the bidlines fixed from Phase 2.

y_i = 1 if dated pairing i is not scheduled to a line crew, 0 o.w.

3.3.3.2 Solution Procedures

Phase 3 combines some of the techniques used from Phase 1 and Phase 2. None of the heuristics are used though, since the problem is small, but the enumeration techniques from both are used as well as delayed column generation. Since the purity rules are important in Phase 3 also, it makes sense to try and solve Phase 3 sequentially in terms of the number of unique pairings in each pattern. The fewer the unique pairings, the more likely it is to generate a pattern that is trip pure. The Phase 3 algorithm is as follows:

Phase 3 MIP Solution Algorithm

Initialize *num_unique_pairings* to 1.

Solve LP relaxation of **Phase 3 MIP** with only columns from Phase 2.

While *num_unique_pairings* ≤ 4

While columns built using *num_unique_pairings* price out to enter basis.

Use column generation to solve the LP relaxation of **Phase 3 MIP**

num_unique_pairings $++$

Solve **Phase3 MIP** using all columns from column generation.

3.4 Conclusion

The circadian rule set outlined in this Chapter addresses many of the concerns related to fatigue for airline crew members. They try to establish and maintain a schedule that will allow pilots to adjust their sleep patterns like any other shiftworker. These

rules along with educating pilots about the causes of fatigue, and ways to mitigate these causes should serve to help reduce the amount of fatigue induced by the 24-hour-a-day operations of the airline industry.

The three phase approach to solving the bidline generation problem reduces at each stage the size of the problem being solved. These smaller problems then should be able to be solved using heuristics to get fast and good solutions. The final solution produced by either Phase 2 or Phase 3 should meet all real world constraints for a working schedule.

Chapter 4

Implementation and Testing

4.1 Implementation

All of the algorithms used to solve the 3 phase approach and bin packing problems were implemented in Microsoft C++ version 6.0 and used the callable libraries of CPLEX version 6.5. All test runs were made on a PC with an Intel Pentium III 500 MHz processor and 384 Megabytes of RAM.

4.1.1 Phase 1

In this section the choice of enumeration versus dynamic programming for solving **COLGEN** directly will be discussed. Then the column generation technique employed will be outlined in detail. Finally, the maximal column heuristic for solving **Phase 1 MIP** is described.

Using a commercial solver to solve **COLGEN** directly was the first consideration. After testing, however, simply adding the best column at each iteration caused slow convergence to the LP relaxation optimal solution, and did not provide enough columns to find a good integer solution when the column generation was finished. A method was needed that was designed to find more than just the best column. As mentioned

earlier, dynamic programming has been used to solve multi-objective knapsack problems, and depending on how the dynamic programming array is stored, all of the good columns can be found quickly in the array.

In trying to solve **COLGEN**, there are six objectives to satisfy; minimum and maximum pay and credit, the number of work days, the number of pairings in a pattern, and the start and end of the rest window. The dimension of this array can be reduced by leaving out the minimum credit constraint by post processing the array and taking only those patterns that meet the minimum credit. It can be further reduced by preprocessing the valid rest windows, and set an indicator if pairing i is a member of rest window r. Finally, the last dimension is the pool of pairings to choose from to form a pattern.

Let $f(i,j,k,l,r)$ be the maximum "utility" achieved by choosing among pairings $1,2,\ldots,i$ with $num_{pairings} \leq j$, $num_{workdays} \leq k$, $num_{credit} \leq l$, and $Rest_{start}$ and $Rest_{end}$ within the rest window r. Then

$$f(m, Max_{pairings}, Max_{workdays}, Max_{credit}, \text{any valid rest window})$$

needs to be computed, where m is the number of pairings and $0 \leq s_i \leq b_i$, for $i = 1,2,\ldots,m$. Using dynamic programming then, the recursion is

$$f(i,j,k,l,r) = \max_{0 \leq s_i \leq b_i} \{\mu_i s_i + f(i-1, j-s_i, k-d_i s_i, l-c_i s_i)\}$$

where all variables are as described in Chapter 3. While this is a feasible way to generate columns, it turns out in this case not to be efficient. The size of the dynamic programming array is

$$num_{pairings} \times Max_{pairings} \times Max_{workdays} \times Max_{credit} \times Total_{restwindows}.$$

In the first test case; $num_{pairings} = 25$, $Max_{pairings} = 7$, $Max_{workdays} = 14$, $Max_{credit} = 4620$, and $Total_{restwindows} = 113$. With these values, the array has more than a billion elements. The actual number of elements can be reduced by preprocessing the actual amounts of credit that are attainable. That is every value of credit from 1 to Max_{credit} is not attainable based on the amount of credit for each pairing. This preprocessing, however, did not change the order of magnitude of the size of the array. With these large dimensions, it is computationally expensive to iterate through all of the possible combinations, not to mention the amount of memory required to store such a large array if you want to be able to choose more than just the best column. It happens that implicit enumeration of the patterns is much more efficient.

During the implicit enumeration the cardinality of a pairing in the pattern is increased while it still meets all of the constraints in equations (4) - (6), and has a valid rest window. If at any time during the building of the pattern, the minimum credit constraint, equation (7), is also met, then the pattern is valid. By sorting the pairings in order of largest to smallest amount of pay and credit, when a pattern fails to meet all of the constraints in **COLGEN** minus the minimum credit constraint, the current pattern is backtracked on by removing one instance of the last pairing in the pattern, and adding one instance of the next smallest pairing in terms of pay and credit, and the pattern generation process begins again. Since pay and credit and the number of workdays are highly correlated, this sorting of the pairings helps reduce the total number of patterns that need to be checked.

The main reason that implicit enumeration is more efficient for this problem, is the fact that the total number of patterns possible is not enormous, while the dimensions

or in this case the righthand sides can be quite large in dynamic programming terms. In all of the test cases the most patterns ever generated was around 400,000. Since all of the patterns are quickly generated, the starting set of patterns that begin the delayed column generation need to be chosen. In general, a simple set of patterns to choose to start the bin packing or cutting-stock problem is the set of maximal columns for each size of final. The equivalent patterns for the Phase 1 problem would be those patterns that use each pairing maximally. Even easier than this is to start with all patterns that use fewer than 3 or 4 different pairings. In practice, this will be less than 5000 columns and many times these are all the columns necessary to find an optimal solution to the LP relaxation of the Phase 1 problem.

Since implicit enumeration is used for the column generation, an efficient way to find the reduced cost of each column is needed. During enumeration, the number of times a pattern has been backtracked is counted and stored with each pattern. Since each subsequent column is similar to the preceding column except for any backtracking done during the enumeration, the reduced cost can easily be calculated and stored in an array by pattern element. Now, the next calculation can be started from the backtracking point instead of starting the calculation over from the beginning. For example, if the first three patterns were

1 1 1 2 2 2,

1 1 1 2 2 2 3, *and*

1 1 1 2 2 3 3.

They would be stored as

$$0\,1\,1\,1\,2\,2\,2$$

$$6\,1\,1\,1\,2\,2\,2\,3$$

$$5\,1\,1\,1\,2\,2\,3\,3\,,$$

where the first number tells us how far in the reduced cost array to start the next calculation. Now, let $Max_{pairings} = 9$, and the dual values be $\mu_1 = 2.5$, $\mu_2 = 1.5$, and $\mu_3 = 4.0$. The reduced cost array for the first pattern would be

$$\begin{bmatrix} 2.5 & 5.0 & 7.5 & 9.0 & 10.5 & 12.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

The next pattern says to add another calculation to the array. So calculation will start with the 12.0 and simply make one addition operation to get 16.0 and add that new element to the array, which would then look like

$$\begin{bmatrix} 2.5 & 5.0 & 7.5 & 9.0 & 10.5 & 12.0 & 16.0 & 0.0 & 0.0 \end{bmatrix}$$

Pricing out the next pattern says to start with the fifth element in the reduced cost array, which is the value 10.5 then two addition operations are made and the new reduced cost array is

$$\begin{bmatrix} 2.5 & 5.0 & 7.5 & 9.0 & 10.5 & 14.5 & 18.5 & 0.0 & 0.0 \end{bmatrix}$$

Using this method, instead of making 17 addition operations, only 8 are made. A significant savings in time when pricing out many patterns.

So, implicit enumeration was chosen as the way to generate columns in the delayed column generation technique. At each iteration, at most 50 columns are added to the problem. It turns out however that with the choice of a starting set of patterns, the

delayed column generation routine was used less than 10 times for any test case.

Once a final set of columns had been found, the integer solution for **Phase 1 MIP** needs to be found. The maximal column heuristic was used here, and as with the bin packing example every column generated was made into a maximal column as well. To use this heuristic, the first concern is the upper bound on a maximal column.

Since a bid period corresponds to a months worth of work, many of the righthand sides of **Phase 1 MIP** are 30 or 31. Using equation (2) gave upper bounds for some patterns in the range from 10 to 30. Thinking about the problem, however, these upper bounds are too high. If a day pure bidline is repeated, the most it can be repeated is 7 times, one instance beginning on every day of the week. Even for a non-day pure bidline, day shifting more than 10 times is difficult. For this problem then the upper bound on a maximal column was chosen to be $k_j = \inf\{7, \text{equation (2)}\}$. With the introduction of the maximal columns, and their associated binary variables, also comes the opportunity to add some constraints to the problem.

Even with the upper bound on maximal columns being held to no more than 7, many of the terms MP_{ij} are greater than half of the righthand side value D_i. When this happens a clique can be built for each pairing. That is, among these variables only one can be positive. This logic of cardinality sets can be continued, not just for one positive variable, but for any integer k.

Let

$$C_{ik} = \left\{ j : \frac{D_i}{k+1} \geq MP_{ij} > \frac{D_i}{k} \right\}, \tag{23}$$

then constraints of the form

$$\sum_{j \in C_{ik}} x_j \leq k \tag{24}$$

can be built. These constraints will be built for each $i = 1, 2, \ldots, m$ and $k = 1, 2, \ldots, 6$. These constraints say that among a set of binary variables, because of the right hand side value and the value of the coefficients of the pattern at each pairing, at most k of the variables in the set may be positive.

Now that a bound on the number of times a pattern is represented in a maximal column has been selected, the value to use for $BONUS$ for each maximal column must be determined. The value here will be determined by how important purity rules are. To give the problem incentive in the minimization, all maximal columns will have a negative coefficient. If a maximal column is trip pure, $BONUS_{n+j} = -k_j$, and if it is not $BONUS_{n+j} = -0.1$. Also, for this problem $c_j = 1$ for all trip pure patterns, and $c_j = 10$ otherwise. Now the final problem can be sent to CPLEX to find a solution.

Even with the maximal columns, **Phase 1 MIP** can still be difficult to solve. An optimal solution in terms of objective value however is not necessary. Instead all that is needed is a feasible solution, and a fixing heuristic can be used to speed up finding a good feasible solution. There are three settings on the fixing heuristic. The first setting turns off the fixing heuristic. The next setting solves the LP relaxation of

Phase 1 MIP with the columns generated as described above. Any maximal column with a solution value of 1.0 is fixed, and then the problem is sent to CPLEX for integer optimization. For the third setting, the LP relaxation is solved using different techniques, which finds different maximal columns at a value of 1.0. These different techniques are iteratively solved and each time any maximal column is found at a solution value of 1.0, it is fixed. This setting usually results in more fixed variables than the second setting, however, it can leave the problem integer infeasible. It may take a couple of runs to determine which if any of the fixing heuristics are necessary for a given instance of the problem. The default used was the third setting as it gave much faster solution times to the Phase 1 problem.

Along with the fixing heuristic, a bounding technique was implemented during the branch and bound portion of CPLEX. Looking at the objective function value at integer feasible points, it is a step function. Any time a maximal column is chosen, there is a step in the value of the objective function. Noting this, when ever an integer solution is found, the next best integer solution can be determined. Usually, a branch and bound technique uses the current best LP solution to prune branches. With this step in the objective function, the upper cutoff for CPLEX can be set to slightly less than the next best integer solution. Doing this eliminates the need to explore any branches whose LP solution is larger than the next best integer solution.

4.1.2 Phase 2

Like Phase 1, enumeration is used to generate all possible feasible bidlines. Again this has proven faster than other techniques such as path search techniques in the DAG formed by the dated pairings. The same pricing scheme as for Phase 1 is used to

quickly price out all of the bidlines generated. Next, the number columns to be added at each iteration of the column generation scheme has to be set. Some techniques add many columns at each iteration, and then remove all but the basic columns or all but the columns with near zero reduced cost. In these techniques, once an optimal LP relaxation solution is found, all columns are priced out one final time, and all columns that meet a minimum reduced cost are added back to the problem for solution. This works well for problems in which the objective function is well defined. This technique does not work well with the Phase 2 problem.

Usually, the objective function in an optimization problem is used to differentiate between the values of different options or variables. The cost of implementing an option, the actual value of a product, or even the amount of waste in an option are all good ways to assign an objective value. For the Phase 2 problem, the goal is to cover all the dated pairings with a specific number of bidlines. This turns the problem from an optimization problem to a feasibility problem. Trying to differentiate between two bidlines is difficult. There is not inherent value of a bidline to optimize, instead you have to use a utility function based on the characteristics of the bidline. The characteristics chosen here were purity rules met, pay and credit, total days off, and the size of the largest continuous block of days off.

Purity was chosen since that is a stated goal of the schedule. The pay and credit levels were chosen to try to meet the requirement to get enough bidlines for all the crews at a crew base. The break points depend on the amount of flying available at a crew base, and the load level per crew desired. Too many bidlines with large credit means fewer crews will be needed, similarly, too many bidlines with small amounts of

credit will means more line crews are needed. Total days off was chosen to force the flying to be done in as few trips as possible. This goes along with the largest block of continuous days off characteristic. If bidlines can be made with 7 or more days off in a row, things like recurring training and vacations can be placed at those times without impacting the whole months schedule. The problem then is to get agreement on the value of trip purity versus day purity, and 14 days off versus 18 days off. Once weights have been assigned to each of the possible outcomes, each bidline can be given a utility by simply adding all of the weights for the characteristic levels it meets. Table 9 shows the 4 characteristics with their 3 levels and weights. The 4 characteristics and 3 levels produce 81 possible combinations, but only 72 different values for the objective function.

Table 9: Bidline Characteristics Used for Utility Function

Purities Met		Pay and Credit		Total Days Off		Largest Block of Days Off	
Type	Val	Amount	Val	#	Val	Days	Val
Perfect	-240	$< Min_{credit} + 120$ mins	-5	≤ 14	1	$\leq$3 days	20
Day	-160	$Min_{credit} + 120$ mins to $Max_{credit} - 120$ mins	-10	15-17	-5	4 to 6 days	-10
Trip	-80	$> Max_{credit} - 120$ mins	-2	$\geq$18	-10	≥ 7 days	-30

The weights chosen for the purity levels were done such that there are four types of bidlines when looking at the objective values. If the objective value is less than -220 then the bidline has perfect purity, if it is between -220 and -141 inclusive it has day purity only and finally if it is between -140 and -61 inclusive it has trip purity only. Any value between -60 and 19 means that the bidline has some characteristic that has been identified, but is not a pure bidline. Now that an objective function can

be applied to the problem, solving the LP relaxation by column generation can be discussed.

It is well known that the LP relaxation of the *set partitioning problem* is highly degenerate. The SIMPLEX method stalls and makes numerous degenerate pivots trying to find the optimal solution to these problems. Even commercial solvers like CPLEX take minutes to solve each iteration, even with a starting basis of the last solution. Knowing this, a way to solve the column generation problem at each iteration quickly is needed. Two methods presented themselves. First, interior point methods have been shown to be very effective in solving these problems. CPLEX's barrier method solved instances of the phase 2 problem in less than 30 seconds, where CPLEX's SIMPLEX solver took more than 8 minutes. Another method that proved useful was Gopalakrishnan's NNLS method [13]. This solution technique unlike SIMPLEX is not restricted to the vertices of the solution space, but can stop on a face of higher dimension. In n dimensions, instead of stopping at a point $(x_1, x_2, \ldots, x_n)$ that is the intersection of at least n half spaces, the NNLS method allows the use of points defined by less than n half spaces. Using these points, the NNLS algorithm does not make degenerate pivots [13]. While test cases were run using the NNLS method, for ease of portability and time, CPLEX's barrier method was used in the final Phase 2 algorithm to solve each iteration of the column generation scheme.

Once the barrier method was chosen to solve each iteration of the column generation, the number of columns to add was determined. As mentioned earlier some methods remove all added columns at each iteration and then do a single pricing out at the end. This did not work for Phase 2. At optimality, too many of the bidlines

Table 10: Pairing Numbers for Reduced Cost Example for Phase 2.

Obj Val	-130	-130	-130	-130	-130	-130	-130	-130	-130	-130	-130
1^{st} Pairing	4	50	96	142	188	280	326	372	418	464	510
2^{nd} Pairing	908	908	908	908	908	908	908	908	908	908	908
3^{rd} Pairing	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200

look similar in terms of reduced cost. Table 10 gives an example of how this happens. Table 10 shows 11 bidlines and their objective values. When only rows 908 and 1200 have been satisfied, these bidlines will all price out differently depending on how satisfied the row is that corresponds to the first pairing. At optimality, if all rows shown in Table 10 are satisfied then the reduced costs of these rows will be zero. If all of these columns are added to the Phase 2 problem at most one of them can be picked in an integer solution. So it may be that not enough other columns for the rows in the first pairing have been added, and once one of these bidlines is chosen then that pairing will be forced to be uncovered. Likewise, if row 908 or 1200 is not satisfied, then its associated dual value will be the big M. This dual value will dominate any other dual values, and again all of the columns in Table 10 will want to enter the problem. One way to fix this problem is to perturb the objective function. A small random number can be added to each objective value to make it slightly different from the rest. To further ensure enough columns for each pairing end up in the Phase 2 MIP, and to give the MIP more room to make choices, at each iteration of the column generation up 1000 columns are added and never removed from the problem. At optimality, some of the columns entered at the first few iterations have a chance of having a large reduced cost, and might help the first condition. Adding 1000 columns seems to be enough to ensure that not just the columns in Table 10 are entered at any iteration, and other columns with the same pairings will also enter.

Once the column generation portion to solve the LP relaxation of **Phase 2 MIP** is finished, the clique based fixing heuristic begins.

For the clique based fixing heuristic the following rule set was used to determine when and which columns to fix. Continuing along with the importance of purity, only perfect and day pure bidlines whose solution value of the LP relaxation is greater than or equal to 0.5 will be fixed. These types of bidlines were chosen as the blocks of days off between each trip leave room for more perfect and day pure bidlines to be built in Phase 3 if necessary. Once the heuristic has finished the fixed Phase 2 problem is solved using CPLEX's MIP solver routine. Here CPLEX is limited to explore only 5000 nodes. Even with the fixed columns, the Phase 2 problem is still a large set partitioning problem and is difficult for CPLEX to solve. By the end of 5000 nodes, the best integer solution is usually within 5% of the best LP relaxation solution, and sometimes, if enough variables are fixed, the fixed problem will fathom to optimality.

4.1.3 Phase 3

In Phase 3 the enumeration techniques of Phase 1 and Phase 2 are combined to find every legal bidline that can be built out of any remaining pairings from Phase 2. The number of bidlines generated in Phase 3 has the potential to be in the millions. The Phase 3 algorithm goes through the bidlines in order of the number of unique pairings used to build the bidlines. At any iteration of the Phase 3 algorithm, if less than 5000 bidlines are generated, all are added to the Phase 3 MIP. When at least 5000 bidlines are generated, the Phase 2 delayed column generation algorithm is used to select the columns added to the Phase 3 MIP. This is continued until all bidlines generated from 4 or less unique pairings are generated or all of the parings are covered by the current

LP relaxation solution. Experience shows that while there are other bidlines that can be generated, their objective value is small and would not make a good bidline for the final schedule.

Once all the columns have been added to the Phase 3 MIP, it is solved using CPLEX's MIP solution routine. Again, the number of nodes to be checked by CPLEX is limited. In Phase 3 this number is 15,000. In many cases, the IP actually stopped before this node limit with a completely fathomed solution.

4.1.4 The Bin Packing Problem

All the techniques discussed for solving the Phase 1 problem were implemented in solving the bin packing problems. There are some slight differences between the two cases however when using enumeration. The biggest difference between bin packing and cutting-stock patterns is the maximality of the pattern. For the cutting-stock problem the large righthand sides dominate the number of times a final can appear in a pattern. For bin packing, this is not always true. The righthand side may be smaller than the number of times the smallest item can be placed in the bin. In these cases, the enumeration scheme for the patterns must be changed. Now patterns are allowed that are not maximal in terms of total measurement. Instead, they will be maximal with respect to the maximum number of times a item can be placed in any one pattern. Another way to account for this difference is to change the equality constraints to greater than or equal to constraints. If an optimal solution exists at equality for all constraints, then the same optimal solution will exist and be found in the relaxed version of the problem as well. The benefit of relaxing the problem is that the problem may be solved using only maximal patterns, and then the patterns

can be post processed to remove any extra items. In this way, the algorithm need not be changed.

4.2 Results

4.2.1 Three Phase Approach to Crew Scheduling

The 3 Phase approach was used on 3 different sets of pairings. The pairings for Test Cases 1 and 2 were generated from 139 flight legs and two crew bases of a short haul F100 fleet. This was not all of the legs for the fleet, but a subset that could be covered by the two crew bases. Each case represents one of the crew bases. All legs were assumed to be flown every day of the bid period. For Test Cases 3 and 4, the actual pairings of a Boeing 757/767 fleet from a major U.S. air carrier were used. The 757/767 fleet was used to ensure that "red-eye" flights would appear in the pairings. The characteristics of each test cases are shown in Tables 11 through 13.

Table 11: Test Case 1 Pairing Data

Pairing Id	Pay and Credit	Block Time per Day				$Rest_{start}$ min's	$Rest_{end}$ min's	Total Days Flown
		1	2	3	4			
0	468	468	0	0	0	1140	2100	30
1	941	369	290	282	0	1140	1754	30
2	611	290	321	0	0	1140	1818	30
3	606	320	286	0	0	1140	1815	30
4	1043	440	299	284	0	1159	1784	30
5	971	342	323	284	0	1391	2100	30
6	573	317	256	0	0	1281	2100	30
7	300	300	0	0	0	1140	2100	30
8	313	313	0	0	0	1140	2100	30
9	953	330	330	293	0	1416	2045	30
10	907	240	302	365	0	1397	2100	30

Table 11: (cont'd)

Pairing Id	Pay and Credit	Block Time per Day				$Rest_{start}$ min's	$Rest_{end}$ min's	Total Days Flown
		1	2	3	4			
11	314	314	0	0	0	1140	2100	30
12	368	368	0	0	0	1140	2100	30
13	316	314	0	0	0	1140	2100	30
14	1060	503	266	272	0	1359	2092	30
15	635	254	381	0	0	1140	1754	30
16	274	274	0	0	0	1140	2100	30
17	361	280	0	0	0	1140	1756	30

Table 12: Test Case 2 Pairing Data

Pairing Id	Pay and Credit	Block Time per Day				$Rest_{start}$ min's	$Rest_{end}$ min's	Total Days Flown
		1	2	3	4			
0	239	239	0	0	0	673	1830	30
1	180	174	0	0	0	634	1845	30
2	194	194	0	0	0	658	1855	30
3	277	261	0	0	0	846	1860	30
4	576	133	396	0	0	553	1784	30
5	840	431	229	128	0	1156	1769	30
6	946	423	175	338	0	1398	2001	30
7	916	328	408	108	0	1209	1810	30
8	1059	233	449	359	0	1245	1898	30
9	360	135	135	0	0	1115	1799	30
10	197	197	0	0	0	1259	2425	30
11	538	123	358	0	0	1173	2129	30
12	918	127	293	395	0	1237	1890	30
13	976	354	259	331	0	1344	2137	30
14	981	251	330	390	0	1401	2090	30

Table 13: Test Cases 3 and 4 Pairing Data

Pairing Id	Pay and Credit	Block Time per Day				$Rest_{start}$ min's	$Rest_{end}$ min's	Total Days Flown
		1	2	3	4			
0	695	325	370	0	0	1235	1940	31
1	656	322	334	0	0	1425	2155	31

Table 13: (cont'd)

Pairing Id	Pay and Credit	Block Time per Day				$Rest_{start}$ min's	$Rest_{end}$ min's	Total Days Flown
		1	2	3	4			
2	1221	397	0	430	394	965	1770	31
3	1448	463	295	343	347	1327	2065	29
4	1440	478	239	338	375	1306	2080	31
5	1450	373	385	356	336	1332	2045	10
6	1450	373	385	356	336	1332	2045	20
7	798	443	355	0	0	956	1815	31
8	1402	369	269	282	475	1396	2250	28
9	1262	321	384	279	278	1370	1785	24
10	1262	321	384	279	278	1370	1785	4
11	706	351	355	0	0	1201	1865	31
12	1057	370	339	348	0	1225	1975	30
13	1427	378	385	347	317	1235	1800	31
14	1373	353	360	314	265	1339	2310	5
15	1373	353	360	314	346	1339	2310	24
16	1373	254	360	314	346	1339	2310	1
17	694	454	192	0	0	1371	1965	31
18	1200	335	394	347	0	1260	1630	1
19	690	325	365	0	0	875	1760	1
20	657	360	297	0	0	1092	1800	1
21	671	299	277	0	0	1247	1870	1
22	596	296	0	0	0	1301	2085	1
23	300	106	0	0	0	1371	2550	1
24	919	315	240	364	0	1670	2295	31
25	1228	295	381	275	277	1660	2365	31
26	1594	219	474	459	421	1165	1785	31
27	1206	285	309	315	297	1590	2235	31
28	1466	464	442	264	296	1639	2085	31
29	1215	395	338	375	0	1706	2255	1
30	900	376	336	0	0	1706	2375	1
31	481	160	0	0	0	1540	2270	1
32	1441	328	295	343	347	1327	2065	1
33	1448	463	295	115	347	1327	2065	1
34	1418	373	248	356	336	1332	2045	1
35	1402	369	269	116	475	1396	2250	1
36	1402	369	269	116	475	1396	2250	1
37	995	95	380	0	0	1270	2040	1

Table 13: (cont'd)

Pairing Id	Pay and Credit	Block Time per Day				$Rest_{start}$ min's	$Rest_{end}$ min's	Total Days Flown
		1	2	3	4			
38	739	370	0	0	0	1330	2160	1
39	1385	300	153	282	475	1396	2250	1
40	715	0	375	0	0	1245	2220	1
41	1249	321	69	279	278	1437	1785	1
42	1249	321	69	279	278	1437	1785	1
43	1267	321	384	0	278	1370	1785	1
44	1080	370	254	348	0	1316	1975	1
45	1373	353	246	314	346	1339	2310	1

All pairings except those for Test Cases 3 and 4 were built using the rest window intersection rules outlined in Chapter 3. Since the pairings in Test Cases 3 and 4 were existing pairings, the rest window intersections were calculated by applying as many of the rules from Chapter 3 as possible. The rest window intersection was chosen to be the largest continuous period of time such that no duty occurred during this time on any days of the pairing. For the bidline generation, Test Case 1 was run with 10, 9, 8 and 0 hour rest window intersections. Test Case 2 was run with 9, 8, 7 and 0 hour rest window intersections since pairings 6 and 7 barely have a 10-hour rest window intersection, and do not share that window with many other pairings. Finally, Test Cases 3 and 4 were run at 5, 4, 3 and 0 hours of rest window intersection, since the smallest rest window for any of the pairings in Test Cases 3 and 4 was 5 hours and 48 minutes.

Since Test Cases 1 and 2 do not have a known solution and a rest window intersection of zero hours does not exclude any patterns or bidlines from being built, all comparisons for these test cases will be made against the solution for the 0-hours rest

window intersection. To account for possible infeasibilities, a goal of 8000 minutes was set for the amount of pay and credit to be allowed to be not scheduled.

The variables shown in Table 3 were set to different levels depending on the data for each case. While most of the values are easily set, Max_{credit} and Max_{fly} must be set with care. In Phase 1 and the pattern generation portion of Phase 3 the actual dates that pairings will be flown on are not known. A bidline that has a multi-day pairing that starts near the end of a bid period may not have all of the days of the pairing actually flown during the bid period. In these cases the actual pay and credit will be less than that calculated for the pattern that generated the bidline. Figure 4 shows an example of how this happens. The first row is a bidline with a 4-day pairing

Days of the Month																															Total Pay and Credit
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
						5	5	5	5				5	5	5	5				5	5	5	5				5	5	5	5	4800
							5	5	5	5				5	5	5	5				5	5	5	5				5	5	5	4500
								5	5	5	5				5	5	5	5				5	5	5	5				5	5	4200

Figure 4: Max_{credit} Calculation Example

repeated 4 times during the bid period. It is a perfect bidline as it is trip and day pure. Each day of the pairing, there is 5 hours of pay and credit. The bidline then has a total pay and credit of 4,800 minutes. If Max_{credit} is set below 4,800 minutes, say at 4,680 minutes, then the pattern used to generate this bidline would not be allowed in Phase 1. By not allowing this pattern in Phase 1, the next two bidlines of Figure 4 would not be generated. Unfortunately, both of these are perfect bidlines

that meet the Max_{credit} rule. To keep from missing what could be good patterns and therefore good bidlines, the value for Max_{credit} in Phase 1 will be increased by the amount of the largest pairing in the test case rounded up to the nearest hour.

For Max_{fly} there are several things that must be considered. First, the FAA requires that no more than 30-34 hours of flying be scheduled for any seven-day period depending on the type of flying operations to be performed. For planning purposes, that would mean a four day pairing would have between $7\frac{1}{2}$ and $8\frac{1}{2}$ block time each day. This would be excessive for any crew. Second, the real world data in Test Case 3 had bidlines that flew almost 27 hours in a seven day period. This equates to a little less than 7 hours of flying each day. A large amount of flying, but manageable. Finally, another maximum can be calculated by taking the maximum between the largest 4 day pairing and twice the largest 2 day pairing in terms of flying hours. In the event that there are no four day pairings, the value for the four day pairing can be replaced by the sum of the largest three day and one day pairings. The minimum of these three values rounded up to the nearest hour was then used for the value of Max_{fly}.

All of the values chosen for each case are shown in Tables 14 - 16. The value for Min_{credit} in Test Cases 1, 3 and 4 was based on the rounding down to the nearest hour the smallest bidline in the known solution of Test Case 3. Max_{credit} will be just ten hours more than Min_{credit} in these cases. For Test Case 2, Min_{credit} had to be reduced to 62 hours as there are pairings that were generated that have very little pay and credit; specifically, pairings 1, 2, 9 and 10. The maximum number of

Table 14: Test Case 1 Parameters

Run Number	Min_{credit}	Max_{credit}	$Max_{pairings}$	$Max_{workdays}$	Max_{fly}	Rest Window Intersection (min's)
1	4080	4680	9	18	26	600
2	4080	4680	9	18	26	540
3	4080	4680	9	18	26	480
4	4080	4680	9	18	26	0

pairings, $Max_{pairings}$, was chosen to be 9 for Test Cases 1, 3 and 4 as it will allow for a 2 day pairing to be flown twice a week, and once during the partial week of a month. While this is type of bidline will be heavily penalized in its objective value for not having many days off, it still can produce perfect bidlines. It is important to note that $Max_{pairings}$ is the only variable that is not associated with a constraint that must be met due to regulations or negotiations. Instead, it is used to limit the number of patterns and bidlines generated. For that reason, Test Case 2 had $Max_{pairings}$ set to 11 as it needed more bidlines to find a solution. $Max_{workdays}$ was set at 18 for Test Case 1 to correspond with the 2-day pairings flown twice each week, while for Test Case 2 it was set at 20 to allow for 5 days of work on each full week of a month. For Test Cases 3 and 4, $Max_{workdays}$ was set at 16 as none of the known solution bidlines had more than 16 days of work in it.

The results for all of the runs are shown in Tables 17 through 19. Table 17 shows the values associated with parameters that can be used to measure how well the quality of life issues were met. Tables 18 and 19 show the varying levels of rest window intersection achieved by the solutions. All of the solutions for Test Case 1 maintained

Table 15: Test Case 2 Parameters

Run Number	Min_{credit}	Max_{credit}	$Max_{pairings}$	$Max_{workdays}$	Max_{fly}	Rest Window Intersection (min's)
1	3720	4680	11	20	26	600
2	3720	4680	11	20	26	540
3	3720	4680	11	20	26	480
4	3720	4680	11	20	26	0

Table 16: Test Cases 3 and 4 Parameters

Run Number	Min_{credit}	Max_{credit}	$Max_{pairings}$	$Max_{workdays}$	Max_{fly}	Rest Window Intersection (min's)
1	4080	4680	9	16	27	300
2	4080	4680	9	16	27	240
3	4080	4680	9	16	27	180
4	4080	4680	9	16	27	0

the number of crews required to cover all the pairings. There was also no significant change in the number of bidlines that met some purity rule. For the other set of short haul flights, Test Case 2, similar results held once the value for Min_{credit} was adjusted to account for the poor pairings generated from the incomplete set of legs.

For Test Case 3 there was a significant difference between the solution found by the airline and that found by the three phase solution methodology. First, the number of crews required to cover all the pairings and the amount of pay and credit left uncovered was less for the three phase approach. Fewer crews means less overhead for the airlines and a savings in personnel costs. Unscheduled flying hours cost the

Table 17: Results for 3 Phase Algorithm

Test Case Run Number	Total Line Crews	Pairings Uncovered	Pay and Credit Uncovered	Perfect Bidlines	Day Pure Bidlines	Trip Pure Bidlines
1.1	72	17	6735	6	52	2
1.2	73	8	3448	17	44	0
1.3	72	16	5973	11	49	1
1.4	72	16	5952	12	46	3
2.1	61	40	9762	24	12	4
2.2	61	39	8966	27	15	14
2.3	61	21	6226	19	23	11
2.4	61	31	7714	17	20	14
3.1	150	3	2394	45	68	19
3.2	149	8	6791	31	84	19
3.3	151	4	3088	39	88	12
3.4	150	5	3744	47	78	11
4.1	148	8	6022	6	79	4
4.2	149	12	12208	44	103	1
4.3	150	7	5500	47	99	2
4.4	150	4	2805	54	94	2
Known Sol.	151	6	4076	7	2	101

airlines more than scheduled flying hours, so the lower amount of pay and credit uncovered also equates to a savings for the airlines. The total number of bidlines that met any purity rule did not change significantly. What is significant is their distribution. There was a remarkable shift from trip pure bidlines to perfect and day pure bidlines. This shift is a direct result of enforcing the quality of life issues during Phase 2 and Phase 3. Test Case 4 showed the same distribution of purities met, and maintained the amount of pay and credit uncovered as well as the number of crews required to cover all the pairings.

Table 18: Rest Window Results for 3 Phase Algorithm: Test Cases 1 and 2

Test Case Run Number	Number of Bidlines with Rest Window Intersection of x hours					
	$0 \leq x \leq 2$	$2 < x \leq 4$	$4 < x \leq 6$	$6 < x \leq 8$	$8 < x \leq 10$	$x > 10$
1.4	0	0	4	7	1	60
1.3	0	0	0	0	4	68
1.2	0	0	0	0	3	70
1.1	0	0	0	0	0	72
2.4	0	0	0	11	14	36
2.3	0	0	0	16	7	38
2.2	0	0	0	0	10	51
2.1	0	0	0	0	4	57

Table 19: Rest Window Results for 3 Phase Algorithm: Test Cases 3 and 4

Test Case Run Number	Number of Bidlines with Rest Window Intersection of x hours					
	$0 \leq x \leq 2$	$2 < x \leq 4$	$4 < x \leq 6$	$6 < x \leq 8$	$8 < x \leq 10$	$x > 10$
Known Sol.	7	7	36	19	42	39
2.4	10	18	3	33	31	54
2.3	0	10	9	69	10	43
2.2	0	0	9	59	40	41
2.1	0	0	11	50	33	56
3.4	10	22	4	32	30	51
3.3	0	10	7	65	13	56
3.2	0	0	5	62	39	44
3.1	0	0	7	48	29	64

4.2.2 The Bin Packing Problem

For this problem, data files from OR Library as well as from the Technische Univesität Darmstadt were used. The test cases from OR-Library come from the instances of the bin packing problem considered in E. Falkenauer (1994) "A Hybrid Grouping Genetic Algorithm for Bin Packing" Working paper CRIF Industrial Management and Automation. The other set of test cases are from the instances considered by A. Scholl, R. Klein, and C. Jrgens (1996): "BISON: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem." [21]

In the first group of test cases, there are 2 classes of bin packing instances. For the first class of problems the following parameters were used

- $n = 120, 250, 500, 1000$
- $c = 150$
- w_j from [20,100] for $j = 1, \ldots, n$

For each of the four instance classes, 20 instances were generated for a total of 80 instances. In all the instances, the data files provided a current best solution from Falkenauer's algorithm. To determine the goodness of the solutions, the lowest possible number of bins was calculated by taking the ceiling of the sum of the weights of all items divided by the capacity of a bin. While this value may not be obtainable, if a solution does obtain it, then that solution is optimal. The algorithm was able to find this value in 79 out of 80 of the instances. In 4 of the instances a solution better than Falkenauer's was found. Table 20 shows the problem identifiers for the 4 solutions that were better than Falkenauer's. Instance u250_13 was the only instance that the solution did not achieve the lowest possible bound. In this case all possible patterns were enumerated, and the ceiling of the objective value of the optimal solution was

Table 20: Solutions for First Class of Falkenauer Bin Packing Problems

Problem Identifier	Lowest Bound	Falkenauer Solution	Maximal Column Solution
u120_08	50	51	50
u120_19	49	50	49
u250_07	103	104	103
u250_12	105	106	105

103 bins. Both the maximal column heuristic and Falkenauer's algorithm found this solution.

The second class of problems has the following parameters

- $n = 60, 120, 249, 501$
- $c = 100$
- w_j from [25,50] for $j = 1, \ldots, n$

The weights in this class are measured in tenths. These problems were generated so that the optimal solution uses exactly $\frac{n}{3}$ bins. Due to machine precision, in many instances the optimal value of the LP relaxation's objective function was within 10^{-6} of $\frac{n}{3}$, but was rounded up to $\frac{n}{3}+1$. For these problems, the maximal column heuristic found a solution of $\frac{n}{3}+1$. Even when the ceiling of the objective function value was $\frac{n}{3}$ the maximal column heuristic was sometimes unable to find an integer optimal solution. Again, a solution value of $\frac{n}{3}+1$ was found. In some instances an optimal integer solution was found.

These results are easily explained however. When the LP relaxation optimal objective

value is less than that of the IP, the column generation adds columns to surpass the IP solution. These columns often times allow the IP to find an integer optimal solution. For problems where the optimal LP relaxation solution value and IP solution value are equal, an optimal solution must be found among just the columns needed to find an optimal fractional solution. In these cases, there is less chance of finding an optimal integer solution. When the problem is restricted to allow only patterns that use 3 or more items, an optimal solution was found in all 80 instances. The files containing the test cases as well as a full listing of the problem areas covered by OR-library can be found at http://mscmga.ms.ic.ac.uk/info.html.

In the second group of test cases, there are three data sets each with a different way of generating the items to be packed. Data set 1 uses parameters

- n = 50, 100, 200, 500
- c = 100, 120, 150
- w_j from [1,100], [20,100], [30,100] for $j = 1, \ldots, n$

The weights are chosen as integer values from the given intervals using uniformly distributed random numbers. For each of the 36 instance classes defined by the settings above, 20 instances are generated resulting in a total of 720 instances. Currently, 704 instances are solved to optimality. The 16 test cases that were not are shown in Table 21 with the possible ranges of an optimal solution and the solution found by the maximal column heuristic. The names of the instances are coded as NxCyWz_v where

- $x = 1$ (for $n = 50$), $x = 2$ ($n = 100$), $x = 3$ ($n = 200$), $x = 4$ ($n = 500$)
- $y = 1$ (for $c = 100$), $y = 2$ ($c = 120$), $y = 3$ ($c = 150$)

Table 21: Results of Instances for Data Set 1

Problem Identifier	Solution Range	Maximal Column Solution
N2C3W2_B	[42,43]	43
N3C2W1_J	[86,87]	87
N3C3W4_B	[87,88]	88
N3C3W4_L	[88,90]	90
N3C3W4_N	[86,87]	87
N4C1W1_M	[245,246]	246
N4C3W4_B	[214,215]	215
N4C3W4_C	[217,218]	218
N4C3W4_E	[218,219]	219
N4C3W4_F	[220,222]	222
N4C3W4_K	[214,215]	214
N4C3W4_M	[215,216]	215
N4C3W4_O	[225,227]	226
N4C3W4_P	[218,220]	219
N4C3W4_R	[213,215]	214
N4C3W4_T	[215,217]	216

- $z = 1$ (for w_j from $[1, 100]$), $z = 2$ ($[20, 100]$), $z = 4$ ($[30, 100]$)
- v =A...T for the 20 instances of each class

The lower bound of the solution range is calculated the same as for the first group of cases. As mentioned earlier, achieving this value guarantees optimality. However, not meeting this value does not negate the possibility that the solution found was in fact optimal. In all solutions, the maximal column heuristic found an integer solution that was equal to the ceiling of the optimal LP relaxation solution value. Since the maximal column heuristic usually runs with a minimum capacity that a bin packing must meet, again optimality cannot be proven by meeting the ceiling of the LP relaxation optimal objective value. The proven optimal solution was found in 2 cases. For

the 11 cases identified by a NxCyW4_v and for N2C3W2_B, the minimum capacity constraint was removed from the heuristic. For all 12 cases, the solution found by the heuristic was in fact optimal.

Data set 2 uses the parameters

- $n = 50, 100, 200, 500$
- $c = 1000$
- $avgWeight = \frac{c}{3}, \frac{c}{5}, \frac{c}{7}, \frac{c}{9}$
- $\delta = 20\%, 50\%, 90\%$

The parameter $avgWeight$ represents the desired average weight of the items, while δ specifies the maximal deviation of the single values w_j from $avgWeight$. For example, the weights are randomly chosen from the interval $[160, 240]$ in case of $avgWeight = \frac{c}{5}$ and $\delta = 20\%$. For each of the 48 classes, 10 instances are generated resulting in a total of 480 instances. Only 3 of these instances have not been solved to optimality. They are listed in Table 22 along with the possible solution ranges and the maximal column heuristics solution. Here again, the values found by the maximal column

Table 22: Results of Instances for Data Set 2

Problem Identifier	Solution Range	Maximal Column Solution
N2W1B2R0	[35,36]	36
N4W2B1R5	[102,103]	103
N4W2B1R6	[101,102]	102

heuristic were equal to the ceiling of the LP relaxation's optimal solution objective value. When the minimum capacity of a pattern was removed, all of these values are

in fact the optimal solution to these instances of the problem.

The third set uses parameters

- $n = 200$
- $c = 100000$
- w_j from [20000,35000] for $j = 1, \ldots, n$

This setting of the parameters guarantees that the weights are widely spread. Furthermore, the number of items per bin lies between 3 and 5. Data set 3 contains 10 instances, only 3 of which are currently solved to optimality. Because the weights are wide spread, these problem better resemble a set covering problem than a bin packing problem. Out of the 10 instances, 3 have all of their righthand sides equal to 1, 6 have only one item chosen more than once, and one has only 3 righthand sides greater than 1. Since these problems do not imitate the Phase 1 problem, they were not solved.

To compare running times of the maximal column heuristic to standard branch and bound and to determine if the heuristic does in fact choose less patterns, each of the instances was also solved using CPLEX's mixed integer optimizer after column generation had been completed, but with no maximal columns added. The times for CPLEX to find an integer optimal solution ranged from seconds to minutes depending on the instance. The number of columns in the problem after column generation determined the time needed by CPLEX to find an optimal solution. Although CPLEX found an optimal solution in most instances as quick as the heuristic, the solutions contained on average 30% more patterns as the maximal column heuristic.

Chapter 5

Conclusions and Recommendations

5.1 Conclusions

The 3 Phase approach to bidline generation is able to build schedules that meet the quality of life issues with no significant increase in cost to the airlines. In all of the test cases, the three phase approach was able to maintain or reduce the number of line crews required to cover all the pairings regardless of the size of the rest window intersection. The number of perfect and day pure bidlines was significantly increased over the known solution. This may however not be a fair comparison, as the airline from which the pairings for Test Cases 3 and 4 were run has changed it's purity emphasis from trip purity to day purity in recent months. However, the total number of pairings that met any of the purity rules was higher for all the runs in Test Cases 3 and 4 when compared to that of the known solution.

The bidlines built were more like shiftwork, and allow pilots to adjust their work-rest schedules just once for the entire bid period. The regularity of the schedule will also allow pilots to plan their off-duty activities in a way to minimize any fatigue as well. While it may be preferable for an airline to maintain trip purity, the perfect and day pure bidlines form better schedules for the pilots.

The purity levels achieved for all four test cases indicate that not only are the number of crews being maintained, but the trade space for doing so does not include purity. There was no significant change in the purity levels for any of the test cases.
Not only were the solutions for the test cases good in terms of coverage and purity, but they also were quick. For Test Case 1, Phases 1 and 2 took less than 15 minutes total, and Phase 3 solved in less than 2 hours. For Test Case 3 the solution times for Phases 1 and 2 were less than 3 minutes total with the entire problem being solved in less than an hour for all instances. Test Case 4 solution times were similar to Test Case 3.

The 3 Phase approach also builds workshifts that are not only weekly and monthly regular, but also kept them daily regular with the rest window intersection rule. Until now, most research into pilot fatigue has dealt with the duration of the duty or rest period. This thesis demonstrates that the actual scheduling of the duty and rest can be affected in such a way as to reduce the fatigue caused by 24-hour-a-day operations.

Finally, the maximal column heuristic used to solve Phase 1 worked equally as well on then bin packing problems. When solving the bin packing problems, the heuristic was able to find the optimal solution in almost every instance. Even when a proven optimal solution was not found, the heuristic found a solution equal to the best-known solution. Not only were the solutions optimal, but they used fewer different patterns. On average, the heuristic chose 23% fewer patterns than simply solving the problem with CPLEX.

5.2 Recommendations

From the results of this work, incorporating a rest window rule set into an actual airline's pairing generator and bidline generator would be the most logical next step. Doing this will allow other rules, unknown to this author, to be tested against the rest window intersection rule. In addition, using a more polished code, should improve the solution times seen in this thesis.

Actually flying a set of bidlines built with a rest window intersection, or polling pilots about their likes and dislikes of this methodology would also be useful. The flying of such schedules would allow data to be gathered as to whether or not a regular schedule and sleep shift significantly decreases the fatigue felt by pilots. Any type of polling or questionnaires would let airlines know how important these and other quality of life issues are to the pilots. Whether or not these issues can be addressed without significant increases in cost to the airlines will not be known until any such study is completed and testing done. The answers, however, will give airlines ideas as to what they can do to improve the quality of life of their pilots, and maintain the safest airlines possible for the public.

As mentioned in Chapter 1, the duty and rest scheduling of reserve crews is also an important aspect of airline crew scheduling. Sohoni et al have developed a reserve crew scheduling technique that builds bidline like schedules for reserve crews to maintain some semblance of a normal schedule [19]. Adding to this idea would be to determine what times scheduled crews have rest window intersections, and try

build bidline like schedules for reserve crews that also contain a fixed rest window intersection. This way reserve crews also would have a given period of time every day of a bid period during which they would never be called to perform flying duties.

Bibliography

[1] Ranga Anbil, E. Gelman, B. Patty, and Rajan Tanga. Recent advances in crew-pairing optimization at american airlines. *Interfaces*, 21(1):62–74, 1991.

[2] Ranga Anbil, Rajan Tanga, and Ellis L. Johnson. A global approach to crew-pairing optimization. *IBM Systems Journal*, 31(1):71–78, 1992.

[3] Cynthia Barnhart, George Nemhauser, Ellis L. Johnson, Martin Savelsbergh, and Pamela Vance. Branch-and-price: Column generation for solving huge integer problems. *Operation Research*, 46(3):51–65, May-June 1998.

[4] Ioannis T. Christou, Armand Zakarian, Jun-Min Liu, and Helen Carter. A two-phase genetic algorithm for large-scale bidline-generation problems at delta air lines. *Interfaces*, 29(5):51–65, Sep 1999.

[5] Vašek Chvátal. *Linear Programming.* W.H. Freeman, 1983.

[6] Chapter I PART 121 Code of Federal Regulations, Title 14. Doc. No. 23634, 50 FR 29319, July 18, 1985, as amended by Amdt. 121-253, 61 FR 2612, Jan. 26, 1996.

[7] Chapter I PART 135 Code of Federal Regulations, Title 14.

[8] Todd E. Combs. *A Tabu Search Approach to Tanker Crew Scheduling.* PhD thesis, Air Force Institute of Technology, 2950 P St, Wright-Patterson AFB OH 45433-7765, July 2002.

[9] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization.* John Wiley and Sons, INC., 1998.

[10] David F. Dinges, R. Curtis Graeber, Mark R. Rosekind, Alexander Samel, and Hans M. Wegmann. Principles and guidelines for duty and rest scheduling in commercial aviation. Technical Memorandum 110404, NASA, May 1996.

[11] Department of Transportation Federal Aviation Administration. Flight crewmember duty period limitations, flight time limitations and rest requirements. 14 CFR Parts 121, 135; [Docket No. 28081; Notice No. 95-18] RIN 2120 - AF63.

[12] S. Folkard. Circadian performance rhythms: some practical and theoretical implications. *Philosophical Transactions of the Royal Society London*, 327(B):543–553, 1980.

[13] Balaji Gopalakrishnan. *Least-Squares Methods for Solving Linear Programming Problems.* PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Ga 30332, June 2002.

[14] R. Curtis Graeber. Aircrew fatigue and circadian rhythmicity. In *Human Factors in Aviation*, chapter 10, pages 305–343. Academic Press, 1988.

[15] Ellis Johnson and Earl Barnes, 2000. ISyE 6679 Georgia Institute of Technology: Computational Methods. Course notes.

[16] Glen Johnson. FAA to begin enforcing crew rest rules. *Associated Press Writer*, June 11 1999.

[17] John M. Meenan. Pilot flight duty regulation and related questions of fatigue, August 18 1999. Testimony at Hearing before the House Transportation and Infrastructure Committee.

[18] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley and Sons, 1988.

[19] Milind Sohoni. *A Robust Optimization Approach to Reserve Crew Manpower Planning in Airlines.* PhD thesis, Proposed, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Ga 30332, 2002.

[20] Donald I. Tepas, Michael J. Paley, and Stephen M. Popkin. Work schedules and sustained performance. In G. Salvendy, editor, *Handbook of Human Factors and Ergonomics, Second Edition*, chapter 32, pages 1021–1058. New York: John Wiley, 1997.

[21] http://www.bwl.tu-darmstadt.de/bwl3/forsch/projekte/binpp/.